UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

# *Semantic and generic object segmentation for scene analysis using RGB-D data*

**by Xiao Lin**

Universitat Politècnica de Catalunya

Departamento de Teoria del Senyal i Comunicacions

# Semantic and Generic Object Segmentation for Scene Analysis Using RGB-D Data

by Xiao Lin

Doctoral Thesis

Advisors: Josep Ramon Casas and Montse Pardàs

Barcelona, March 2018

# Abstract

In this thesis, we study RGB-D based segmentation problems from different perspectives in terms of the input data. Apart from the basic photometric and geometric information contained in the RGB-D data, also semantic and temporal information are usually considered in an RGB-D based segmentation system.

The first part of this thesis focuses on an RGB-D based semantic segmentation problem, where the predefined semantics and annotated training data are available. First, we review how RGB-D data has been exploited in the state-of-the-art to help training classifiers in semantic segmentation task. Inspired by these works, we follow a multi-task learning schema, where semantic segmentation and depth estimation are jointly tackled in a Convolutional Neural Network (CNN). Since semantic segmentation and depth estimation are two highly correlated tasks, approaching them jointly can be mutually beneficial. In this case, depth information along with the segmentation annotation in the training data helps better defining the target of the training process of the classifier, instead of feeding the system blindly with an extra input channel. We design a novel hybrid CNN architecture by investigating the common attributes as well as the distinction for depth estimation and semantic segmentation. The proposed architecture is tested and compared with state-of-the-art approaches in different datasets.

Although outstanding results are achieved in semantic segmentation, the limitations in these approaches are also obvious. Semantic segmentation strongly relies on predefined semantics and a large amount of annotated data, which may not be available in more general applications. On the other hand, classical image segmentation tackles the segmentation task in a more general way. But classical approaches hardly obtain object level segmentation due to the lack of higher level knowledge. Thus, in the second part of this thesis, we focus on an RGB-D based generic instance segmentation problem where temporal information is available from the RGB-D video while no semantic information is provided. We present a novel generic segmentation approach for 3D point cloud video (stream data) thoroughly exploiting the explicit geometry

and temporal correspondences in RGB-D. The proposed approach is validated and compared with state-of-the-art generic segmentation approaches in different datasets.

Finally, in the third part of this thesis, we present a method which combines the advantages in both semantic segmentation and generic segmentation, where we discover object instances using the generic approach and model them by learning from the few discovered examples by applying the approach of semantic segmentation. To do so, we employ the one shot learning technique, which performs knowledge transfer from a generally trained model to a specific instance model. The learned instance models generate robust features in distinguishing different instances, which is fed to the generic segmentation approach to perform improved segmentation. The approach is validated with experiments conducted on a carefully selected dataset.

# Acknowledgments

I would like to acknowledge the help, support and guidance of my thesis advisors, Professor Josep Ramon Casas and Professor Montse Pardàs, with the most sincere gratitude. They offered me the freedom to work on the research subject I like and explore it, always encouraged me to keep trying when I wanted to give up. I could not have imagined having a better advisors and mentors.

I am very grateful to my groupmates in the Image Processing Group (GPI) of Signal Theory and Communications Department (TSC) of UPC for the help and support in all the time of research. I also want to acknowledge the technical support from Albert Gil Moreno and Josep Pujal.

I am indebted to my parents and my loving wife, Yi Sun, for their always unconditional support and encouragement. Their love has provided me with the enthusiasm to rise to the challenges and conquer them.

Finally, I want to thank the finance support from the Joint China Scholarship Council (CSC)-Universitat Politècnica de Catalunya (UPC) Scholarship and the projects BIGGRAPH (TEC2013-43935-R) and MALEGRA (TEC2016-75976-R).

# Contents

# List of Figures

# Acronyms

**ACM** Active contour model. 25

**AR** Augmented Reality. 43

**ARD** Absolute Relative Difference. 48, 50, 54

**ASMS** Adaptive Surface Models based 3D Segmentation method. 86

**ASPP** Atrous Spatial Pyramid Pooling. 53, 106

**BOV** Bag of Visual Words. 17

**BPT** Binary Partition Tree. 30

**C** Class Average Accuracy. 47, 52, 56, 57

**CNN** Convolutional Neural Network. iii, 10–12, 16, 20–23, 28, 31, 33–35, 59, 97–99, 102, 103, 108, 110–112, 114–116, 119, 122, 123

**CRF** Conditional Random Field. 67, 68, 73, 77, 82, 90, 94, 114, 125–128

**DeepLab-ASPP** Deeplab-Atrous Spatial Pyramid Pooling. 37–39, 42, 45–47, 52–54, 104, 106

**DMMS** Dynamic Management of Merge and Split. 82

**FCN** Fully Convolutional Network. 21, 23, 45, 53

**FPFH** Fast Point Feature Histograms. 18, 64

**G** Global Accuracy. 47, 52, 57

**HOD** Histogram of Oriented Depth. 18

**ILSVRC** ImageNet Large-Scale Visual Recognition Challenge. 122, 124

**LRN** Local Response Normalization layer. 40, 41

**mIoU** mean Intersection over Union. 47, 52, 57

**PFH** Point Feature Histogram. 18

**PLEDL** Pixel Level Enconding and Depth Layering. 23

**PP** Percentage of Pixel. 48, 54

**PP-MVN** Mean Variance Normalized Pixel of Percentage. 48, 54

**ReLU** Rectified Learnear Unit. 40, 41, 120, 121

**RMSE-linear** Linear Root Mean Square Error. 48, 54

**RMSE-log** Log Root Mean Square Error. 48, 54

**RNG** Random Number Generator. 42

**SGD** Stochastic Gradient Descent. 107, 122

**SIE** Scale Invariant Error. 48, 55

**SIFT** Scale Invariant Feature Transform. 17, 118

**SLIC** Simple Linear Iterative Clustering. 27

**SRD** Square Relative Difference. 48, 50, 54

**SUN-RGBD** RGB-D Scene Understanding Benchmark Dataset. 51

**SVM** Support Vector Machine. 17, 19

Chapter 1

# Introduction and Overview

A visual scene is commonly defined as a view of an environment composed of objects organized in a meaningful way, like a kitchen, a street or a forest path. More broadly, the domain of scene perception includes any visual stimulus that contains multiple elements arranged in a spatial layout, for example a shelf of books, an office desk, or leaves on the ground. When a human vision system perceives a scene, several abilities are involved in the process, such as separating elements in the scene, identifying different elements or constructing spatial/temporal relationships between elements [Gol10].

In a computer vision system, a scene is information that flows from a physical environment into a perceptual system via sensory transduction [RB94, Gei08]. The usual information input for computer vision tasks is a color image. Color images capture a projection of the scene into the image plane with discrete values at each sampled pixel providing a dense color representation. The projective nature of imaging systems loses one dimension of the scene geometry. Therefore, adding the distance from a scene to the camera at each pixel brings back a highly informative feature. The distance information provides the 3D geometry, object poses and spatial layout of the projected scene. The distance information is usually encoded in a depth image, registered with a color image. A video formed as a sequence of color+depth images is also usually leveraged as a type of input. It provides the dynamics of a scene, that is to say, the temporal information of a scene is available for both photometry (color)

and geometry (depth). A color image and the registered depth image are paired as an **RGB-D frame**. A video containing RGB-D frames is known as an **RGB-D video/stream**. Mimicking the abilities of human perception in processing multiple types of information, computer vision systems aim to endow machines with smart perception skills. In practice, several tasks in computer vision, such as segmentation, object detection and object recognition, usually deal with information fusion in some way.

In this thesis, we focus on solving segmentation problems with RGB-D data in a computer vision system. Segmentation is an essential task serving as the foundation for higher level problems such as object recognition and scene analysis. Regarding the involved information, we can categorize segmentation problems into several classes. In the following sections, a brief introduction for different segmentation problems are described, while the advantages and challenges of exploiting RGB-D data in such problems are discussed. In Section 1.1, we define static image segmentation tasks taking only photometric information as the input (RGB based) and those based on both photometric and geometric information (RGB-D based). In each case, we further categorize into unsupervised/supervised segmentation. Section 1.2 defines a video segmentation task and how RGB-D can help.

## 1.1 Static Image Segmentation

Image segmentation is an ill posed problem, since a unique solution does not exist. Fig.1.1 shows an example illustrating that different segmentation exists when performing manual segmentation of a color image. The criterion of a segmentation system varies regarding the final application.

### 1.1.1 RGB based Image Segmentation

In general, RGB based image segmentation is defined as a task in which the segmentation system $F$ takes a color image $I$ as input and outputs a segmentation

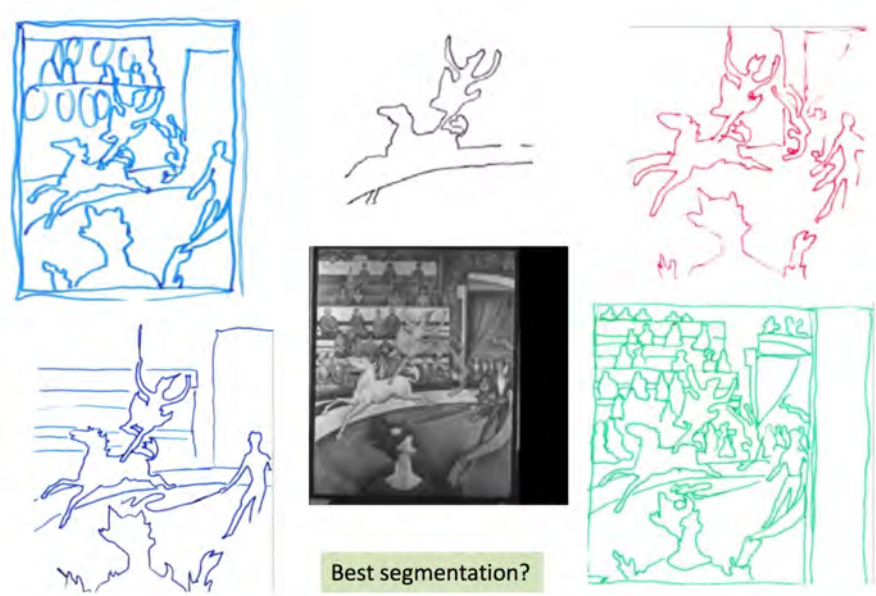



Figure 1.1: An example of the manual segmentation result of a color image

mask $L$, $F(I) \rightarrow L$. The segmentation mask represents segments of the color image using different labels.

### RGB based Unsupervised Image Segmentation

Image segmentation is traditionally defined as partitioning the image into a set of segments showing some sort of pixel homogeneity (known as **unsupervised segmentation** or **low level segmentation**). The connected labels $l_i$ of the segmentation mask $L$ create a partition $\{l_i\}$ of the image $I$, so that $\cup l_i$ is the complete image support and $l_i \cap l_j = \emptyset$ for all $i \neq j$. Usually, the labels optimize some segmentation criterion $C$, so that $C(l_i) = True$ for all $i$, and $C(l_i \cup l_j) = false$ for all $i$, $j$. Connectivity can be understood here as $4-$ or $8-$ connectivity over an square image grid.

In unsupervised segmentation approaches, pixels are usually grouped into segments according to the criterion $C$ and regarding to their local homogeneity. The segments obtained in such low level segmentation approaches are more perceptually meaningful than raw pixels, while also producing a simplified representation of the image, which can be exploited by higher level segmentation and classification approaches. However,

obtaining high level segmentation such as object segmentation only based on unsupervised approaches is extremely difficult due to the lack of semantic information to define the segmentation criterion.

### RGB based Supervised Image Segmentation

In order to produce more meaningful regions, which *ideally* correspond to semantic objects in the scene, high level supervision is usually incorporated into the process. Supervised image segmentation methods are based on image models or classifiers able to introduce semantic hints into the segmentation criterion. Semantics are introduced as a classification strategy defined through training from semantically enriched data elaborated or supervised by humans, such as predefined object models or annotated data. According to the supervision used in the methods, they are further categorized, such as model based segmentation, semantic segmentation, etc. Among them, **semantic segmentation** is one of the hottest fields in the last decade. Semantic segmentation aims to recognize the semantic category of the image at the pixel level.

A classifier $\Theta$ is usually trained on a limited number of classes/semantics $\theta$, in order to output the class label of each pixel $L(x, y)$ $(\Theta(I, x, y \mid \theta) \rightarrow L(x, y))$. $x$ and $y$ denote the coordinates of the pixel in the image. The pixel classes resulting of a classification are, in principle, not necessarily connected, but a different label may be assigned to each connected component to get a segmentation partition that fulfills the definition above. However, training the classifier requires a large amount of annotated data, which may not be available in some applications. Besides, semantic segmentation restricts image segmentation to a few types of semantics, which makes it difficult to scale this supervised segmentation approach to more generic applications.

One other problem of semantic segmentation is that it generally produces class-aware labels for a scene without being aware of individual object instances. To distinguish individual object instances, **instance segmentation** approaches, such as [GGAM14, SSF14, HGDG17] are proposed. The idea of instance segmentation is to identify the different object instances for the same category label. The segmentation at instance level provides a better foundation for higher level applications than the

raw output of a pixel level classifier.

### 1.1.2 RGB-D based Image Segmentation

The widespread availability of RGB-D data from consumer depth sensors provides the possibility to work with explicit 3D geometry, i.e. point clouds in 3D space with physical coordinates X-Y-Z where true distances can be measured, instead of pixels $(x, y)$ in the 2D image plane for which only projective distances are available. Consumer depth sensors like Kinect, Asus Xtion, Realsense or Orbecc sensors, capture a color image registered with a depth image. This can produce a 3D point cloud by transforming the per-pixel distances provided in the depth image using camera parameters. The richer information from actual 3D geometry data in the real world can be exploited to improve segmentation. RGB-D based image segmentation considers both photometric and geometric information to solve image segmentation tasks, in which the segmentation system $F$ takes a color image $I$ and registered depth map $D$ as the input, and outputs a segmentation mask $L$, $F(I, D) \rightarrow L$.

#### RGB-D based Unsupervised Image Segmentation

Large attention has been drawn on RGB-D based unsupervised methods, which exploit only generic features for segmentation. Both photometric from the color image, and geometric features like 3D connectivity and 3D shape are taken into account to help defining the criterion in a segmentation task. The physical attributes of the 3D geometry information ease the unsupervised segmentation problem. True 3D distances in point clouds provide another cue to sense object boundaries in the scene, particularly where occlusion contours are located. Modeling the scene with physically meaningful 3D features in the real world also helps configuring parameters for real applications (i.e. human height is not something around $x$ pixels anymore, but a true dimension of about $1.70m$ in the actual data). This fact may boost the performance of unsupervised approaches for RGB-D data with respect to RGB data.

The genericity of unsupervised approaches on RGB-D could also serve as a building

block for more powerful supervised approaches in the future. On the other hand, new challenges also emerge for RGB-D data, as the depth information is usually noisy, sparse and unorganized, which makes the strategy of analyzing 3D point clouds or extracting generic 3D features critical in RGB-D based approaches.

### RGB-D based Supervised Image Segmentation

Supervised image segmentation approaches usually introduce semantics to help defining the criterion in a segmentation task by using annotated data. The lack of annotations for RGB-D datasets compromises the application of label-hungry supervised segmentation methods for RGB-D data. For instance, NYU depth v2 dataset [NSF12] contains only 1449 pairs of aligned RGB and depth images. SUNRGB-D dataset [SLX15] contains more annotated data (10355 pairs), but it is not yet comparable to the figures of annotated data for RGB datasets like ImageNet [DDS+09], MS-COCO[LMB+14]. Besides, the annotated RGB-D datasets are usually restricted to indoor scenes due to the technical limitation of consumer depth sensors, such as limited range for depth sensing and poor robustness to ambient infrared noise in outdoor scenes.

The lack of annotated RGB-D datasets, usually requires data augmentation techniques (i.e. flipping or cropping), in order to synthesize enough annotated data for training in supervised RGB-D segmentation. A different way to exploit depth in a supervised segmentation system is proposed in multi-task learning schemes [EF15, MPK16], in which the system is trained to jointly estimate the depth image and a segmentation mask for a color image. Since depth estimation and semantic segmentation are two strongly correlated tasks, addressing them into unified approaches can be mutually beneficial. In this case, depth maps acquired from the depth sensor are used for training the system, in order to improve segmentation results even when depth will not be available at the testing stage.

## 1.2 Video Segmentation

A video formed as a sequence of images is usually employed in applications such as video surveillance and human motion analysis. Intuitively, video segmentation can be treated separately for each frame as a set of static image segmentation tasks. However, the spatial segmentation in each of the video frames is not always temporally consistent in the whole sequence due to the lack of necessary coherence constraints along the sequence. Therefore, it is worthwhile to consider temporal information in video segmentation straight from the start, instead of considering temporal coherence just as a constraint or as a post-processing issue.

Exploiting temporal information usually comes down to establishing the temporal correspondences between consecutive video frames. The correspondences can be established at different levels. For instance, Optical Flow techniques compute temporal correspondences at pixel level, estimating the pixel motion between to frames. Building temporal correspondences at pixel level is a difficult task. For video segmentation purposes, often suffices to briefly represent local raw pixels at some higher scale and then refine this representation at each frame by analyzing temporal correspondences, in order to provide a temporally coherent segmentation.

In the following sub-sections, we first explain how the video frame representation problem is generally defined (Section 1.2.1). Then, we introduce the definition of building temporal correspondence based on the representations (Section 1.2.2). We also discuss how depth information can help in these two tasks.

### 1.2.1 Video Frame Representation

Representing video frames aims at grouping a large number of pixels into a small number of segments, while the object boundaries should be well preserved. In this manner, this brief representation of video frames simplifies the problem of building temporal correspondences. Representing video frames has a similar goal to the task of static image segmentation introduced in Section 1.1. In this case, more complex video frame representations such as hierarchical representation, generic object models

or object proposals are proposed in order to provide higher flexibility when building temporal correspondences. In the following subsections, we briefly review RGB based video frame representation methods, and explain how RGB-D data can help from this perspective.

### RGB based Video Frame Representation

The introduction of a hierarchical representation allows establishing correspondences at different scales, which benefits the temporal coherence analysis. A hierarchical representation describes the raw data from coarse to fine, usually starting from segments at a relatively fine level generated by over-segmentation. Then, the segments are gradually grouped into coarser level regions.

Generic model-based representations describing objects in the scene with models such as surface model or Gaussian mixture model are employed to incrementally learn/update an object model along a sequence, which allows better tracking, or building temporal correspondences. But these object level representations still rely on a good initial configuration for the model. Since they are incrementally updated in each frame, they are also very sensitive to segmentation errors, which may accumulate over time.

Supervision is also usually exploited to represent a video frame. Apart from the supervision employed in static image segmentation methods, which can be leveraged to video frame representation, initialization may also serve as one special type of supervision required in some tracking based video segmentation methods [RM07, TFNR12, PKB+17, CMPT+17]. The advantage of introducing initialization is obvious, as it provides clear targets/models for the system to perform robust segmentation in the following video frames. But it restricts these methods to certain types of application scenarios, where only a few foreground objects are of interest. Most computer vision applications involve large amounts of data with different types of scenes containing several objects, and this requires more genericity in video segmentation.

More recently, a representation based on object proposals [LKG11, ML12, ZJS13] have been introduced to represent raw data with a pool of object-like regions. These

'object proposals' are extracted from each frame based on generic spatial features. The advantage of these methods is that the extracted object proposals usually corresponds to object level segments. But to cover all the objects in the scene in the object proposal pool, redundant proposals are inevitably generated, which makes it computationally expensive.

#### Representation based on RBGD data

When RGB-D stream data is available, the added depth information makes more efficient to represent raw data in video frames. For instance, in hierarchical representations, depth information provides a better over-segmentation by considering the 3D spatial relations between pixels. It also helps defining a better grouping strategy when building the hierarchy. In generic model-based approaches, depth information provides the possibility to define the generic model in 3D rather than 2D. In representations based on object proposals, depth information, serving as an extra generic feature, helps on better generating the proposals. In supervised approaches, depth information provides an extra type of input sources so that richer features can be designed or learned in the training process to improve the performance of classifiers.

### 1.2.2 Building Temporal Correspondences

A video formed as a sequence of images also provides temporal information between elements in those images. Exploiting temporal coherence offers a way to segment video frames spatio-temporally. Temporal correspondences between video frames are built at different levels depending on the representation with varying difficulty at each level. The difficulty of building temporal correspondences varies depending on different levels. Temporal correspondences at the finest (pixel level) generated between consecutive frames implicitly represent the optical flow of the two frames. In this case, the global optimum in the correspondence building task is still difficult to achieve considering the large scale of the problem. Instead, a local optimum is usually accepted as a solution for the correspondence problem, which makes the established

correspondences less reliable in subsequent analysis. On the other hand, building temporal correspondences at a coarser level reduces the scale of the problem, which allows to search for the global optimum in the correspondence building task. However, the segments at the coarser level between consecutive frames are not always temporally consistent, which may also lead to errors when building temporal correspondences.

When RGB-D stream data is available, the temporal correspondences between objects in consecutive frames may show the actual movements/displacements of objects in the real world 3D space. That is to say, building temporal correspondences for RGB-D stream data can be modeled based on a clear physical meaning. For instance, 3D displacements are more reliable than 2D displacements due to the fact that objects are captured at different scales on 2D images.

As explained in the previous sections, the emergence of depth data provides another source of data which can be applied in almost all aspects to better approach segmentation problems in computer vision. However, the problem of how to incorporate efficiently depth information in segmentation tasks still needs further study.

## 1.3 Scope and Goals of This Dissertation

In this thesis, we aim to address different segmentation problems with RGB-D data. Despite the huge progress the field has experienced in the last few years, we consider there is still room for improvement. Since this dissertation has been performed in a period of fast changes in the state-of-the-art techniques, we investigate the way to efficiently incorporate depth information for both unsupervised segmentation problems and supervised segmentation problems.

Semantic segmentation is a core problem in the field of supervised image segmentation. With the successful application of Convolutional Neural Networks (CNNs) to semantic segmentation, a huge progress has been made in this field for RGB data. In this thesis, we first address the semantic segmentation problem by introducing depth information into state-of-the-art CNNs. We incorporate depth information in the training process of the CNN by following a multi-task learning framework.

Although CNNs based semantic segmentation methods achieve outstanding segmentation result due to the strong representation power of CNN, there are still some drawbacks which may limit its application for higher level applications. CNN based semantic segmentation restricts the approaches to several predefined semantics, while it usually requires a large amount of annotated data for training a classifier in order to obtain semantic labels at pixel level. Apart from that, semantic segmentation is also not aware of object instances, which makes it hard to apply to instance level applications. From the other perspective, most of the unsupervised image segmentation approaches naturally have the advantage on coping with generic scenes. However, object level segmentation can hardly be achieved in those approaches due to the lack of semantic information. To tackle these problems, we address an unsupervised generic instance segmentation problem based on RGB-D stream data, in which spatial-temporal information are analyzed based on generic features extracted from RGB-D data.

The performance of the generic instance segmentation method is highly restricted to the discriminative power of the employed hand-crafted features. On the other hand, CNNs based semantic segmentation methods introduce a good representation for the predefined semantics, which are trained to extract robust features via networks with a huge number of parameters. However, the cost of training those CNNs is not affordable in generic instance segmentation. In these situations, we propose a method to combine the genericity of generic instance segmentation and the strong representation power of CNNs by employing the idea of one shot learning which learns an object model based on one (or very few) example of an object discovered in a sequence.

Our work and contributions are divided into three main parts listed hereafter:

**Part I: Semantic Segmentation based on RGB-D Data**

In this part, we address the problem of how to incorporate depth information in a CNN based semantic segmentation task.

Our contributions are:

- A novel hybrid CNN architecture for jointly tackling depth estimation and se-

mantic segmentation, in which both tasks benefit from each other in the hybrid architecture

- We clarify how the two tasks help each other in a hybrid system by investigating the common attributes as well as the distinction in depth estimation and semantic segmentation

- The hybrid architecture is verified and applied in different scenarios

**Part II: Generic Instance Segmentation based on RGB-D Stream Data**

Semantic segmentation is restricted to a few predefined types of semantics, which can be hardly applied to more generic scenes, such as when undefined new objects move into the scene. On the other hand, semantic segmentation is not aware of individual object instances. This also compromises the application of semantic segmentation for applications such as interaction analysis and instance counting. In this part, we propose an unsupervised instance segmentation approach based only on generic features extracted from RGB-D stream data.

Our contributions are:

- A novel hierarchical representation for the 3D point cloud of a scene, which allows to establish temporal correspondences efficiently at different scales of object-connectivity

- An approach to tackle the temporal correspondence problem. The proposed approach converts the problem to a labels assignment task, and solves it with an optimization method

- A mechanism to deal with possible object splits and merges along time. It maintains the similarities between nodes in the hierarchy in order to deal with possible object splits and merges.

**Part III: One-Shot Learning for Generic Instance Segmentation based on RGB-D stream data**

Generic features employed in generic segmentation methods have limited discriminative power to distinguish different objects in the scene, while CNN based semantic segmentation is restricted to predefined semantics and not aware of object instances. In this part, we combine the advantages of the two methods, that is the strong object representation power of CNNs and the genericity of the unsupervised instance segmentation method, and apply the combined approach to solve a generic instance segmentation problem in RGB-D video sequences.

Our contributions are:

- A one-shot learning approach, allowing to represent the appearance of an object instance when it is discovered by the generic instance segmentation method

## 1.4 Organization of The Thesis

Specific contributions of this thesis mentioned in the three parts above are discussed in each corresponding Chapter (3, 4, and 5) after the relevant state-of-the-art (Chapter 2). The conclusions in Chapter 6 summarize the contributions, and also include a list of journal papers, international conference publications, contributions to projects and submitted publications as a result of the work in this thesis. The basic knowledge related to the thesis is included in the appendices. The organization of this thesis is, thus, as follows:

**Chapter 1** Introduction to the thesis subject, the background of this subject and the organization of this thesis.

**Chapter 2** An overview of the relevant state-of-the-art publications and methods related to semantic segmentation and generic segmentation.

**Chapter 3 (Part I): Semantic Segmentation based on RGB-D data**

**Chapter 4 (Part II): Generic Instance Segmentation based on RGB-D Stream Data**

**Chapter 5 (Part III): One-Shot Learning for Generic Instance Segmentation based on RGB-D stream data**, where classical generic segmentation and semantic segmentation techniques are combined.

**Chapter 6: Conclusions and Future Work** The final conclusions, contributions, related publications and future work are included in this chapter.

**Appendix A** The fundamentals of Convolutional Neural Network related to this thesis.

**Appendix B** The fundamentals of Conditional Random Field related to this thesis.

Chapter 2

# State of the Art

Segmentation is a classical topic in computer vision. The computer vision community has produced an increasing number of contributions to address this problem in recent years. Traditionally, the segmentation techniques were developed based mainly on photometric information in color images. The recent emergence of consumer depth sensors provides the opportunity to incorporate geometric information in the segmentation process coping with problems that could hardly be addressed only based on photometric data, such as scale changes, lighting conditions and background clutter.

Following the taxonomy introduced in Chap 1, we have categorized segmentation problems into RGB/RGB-D based segmentation, unsupervised/supervised segmentation and image/video segmentation. In this thesis, we aim at solving different segmentation problems with RGB-D data. Specifically, we start with a semantic image segmentation problem, which is core in supervised segmentation. We study the current background of approaches tackling semantic segmentation problems in Section 2.1. Complementary to semantic segmentation, unsupervised image segmentation systems can be applied in more general applications where the semantics cannot be predefined, or instance level segmentation is needed. We review these approaches in Section 2.2. Unsupervised approaches mainly focus on generating spatially coherent segments and can hardly produce segments at object level due to the lack of information in a single image. However, they provide a good image representation for further analysis when more information is available, such as temporal information

in video sequences. Exploiting temporal information may help in achieving object level segmentation in unsupervised approaches. Thus, we study video segmentation approaches in the state-of-the-art by reviewing methods exploiting temporal information in Section 2.3. In each category, we first review RGB based methods in the state-of-the-art, then explain how depth information helps when these methods are extended to deal with RGB-D, so that we study the contribution of RGB-D data in different segmentation problems.

## 2.1 Semantic Segmentation

Semantic segmentation is a task of recognizing and understanding object classes at the pixel level. It follows the schema of recognition, in which pixels are first represented by features, then classified into different object classes. In the following subsections, we will first review the traditional semantic segmentation techniques, then we study the more up-to-date Convolutional Neural Networks (CNNs) for semantic segmentation. Under each branch, we review both RGB and RGB-D based approaches to study how RGB-D data helps in different situations.

### 2.1.1 Traditional Semantic Segmentation

Traditionally, semantic segmentation is done with a classifier which operates on fixed-size feature inputs and a sliding-window approach. The classifier is usually trained on features extracted from image patches with a fixed size. In the test phase, the trained classifier is fed with features extracted from rectangular regions of an image, which are called windows. The center pixel is classified according to the information contained in the window. Sliding the window and classifying its features produces the labels for all pixels on the image. The two main problems in classical semantic segmentation approaches are: 1) pixel description, in which we study how to represent a pixel on the image with features in order to train the classifier efficiently, 2) training techniques, in which we study methods used to train a classifier based on

the extracted features.

### RGB based Pixel Description

The choice of features is very important in traditional approaches. In this section, we review the most commonly used features in a semantic segmentation task.

**Pixel Color:** Pixel color is the most widely used feature. Pixel color is described in different color spaces, providing different representations. Typically the colors of pixels in an image are represented in the RGB color space. No single color space has been proven to be superior to all others in all contexts [CJSW01]. However, the most common choices seem to be RGB and color spaces based on luminance and color.

**Histogram of Oriented Gradient:** The original image is transformed into two feature maps of equal size which represent the gradient, that is, the partial derivative in $x$ and $y$ for each pixel. These feature maps are split into patches and a histogram of the directions is calculated for each patch. HOG features were proposed in [DT05] and are widely used for segmentation tasks [BMBM10, FGMR10].

**Scale Invariant Feature Transform (SIFT):** SIFT descriptors [Low04] represent key-points in an image. An image patch of size $16 \times 16$ around the key-point is taken. This patch is divided in 16 distinct parts of size $4 \times 4$. For each of those parts a histogram of 8 orientations is calculated similar as for HOG features. This results in a 128-dimensional feature vector for each key-point. In [PTN09], Plath et al. use SIFT to describe image patches at different scales and train the classifier on patch features using a Support Vector Machine (SVM) to produce semantic labeling.

**Bag of Visual Words (BOV):** BOV is based on vector quantization. Similar to HOG features, BOV features are histograms that count the number of occurrences of certain patterns within a patch of the image [CDF$^+$04]. In [CP08], Csurka et al. use BOV to transform the low level features into a high level representation. A Gaussian Mixtured Model (GMM) is employed to model the visual vocabulary of the low level features, in which each Gaussian corresponds to a visual word. These high level features are then used for semantic segmentation.

### RGB-D based Pixel Description

Consumer depth sensors produce a depth map registered with a color image, where distance information of the scene to the camera is stored pixel-wise. In the case of feature extraction on a depth map, a straightforward way is to extend the existing methods for color and apply them to depth maps. For instance, Histogram of Oriented Depth (HOD) [SA11] is proposed similarly to HOG, in which a histogram of the orientation of the depth gradient is calculated for patches on a depth map. Besides, a 3D point cloud can be back-projected from a depth map by transforming the per-pixel distances using the camera parameters. The 3D point cloud represents discrete points of the surface of the scene from the viewpoint of the camera, with rich geometric information. Several 3D features extracted from 3D point clouds are proposed.

**3D Coordinates:** 3D coordinates of a point on a point cloud represents its spatial localization in the real world 3D space. The similarity between 3D coordinates of different points shows their physical spatial relationship.

**3D Normals:** 3D normals are important features of a geometric surface, and are also used to create higher level 3D features. Given a geometric surface, it is usually trivial to infer the direction of the normal at a certain point by estimating the vector perpendicular to the surface at that point. Since the point cloud acquired from consumer depth sensors represents a set of points sampled on the real surface, there are usually two ways to approximate the surface normal of a point:

- obtaining the underlying surface from the point cloud, using surface meshing techniques, and then compute the surface normals from the mesh
- using approximations to infer the surface normals from the point cloud directly

For instance, in [Rus09], estimating the surface normal is reduced to an analysis of the eigenvectors and eigenvalues of a co-variance matrix created from the nearest neighbors of the query point on the point cloud.

**Point Feature Histogram (PFH)/Fast Point Feature Histograms (FPFH):** The idea of the PFH [Rus09] is to encode the geometrical properties of the $k$-neighborhood of a point by generalizing the mean curvature around the point from a multi-dimensional

histogram of values. This highly dimensional hyperspace provides an informative signature for the feature representation, is invariant to the 6D pose of the underlying surface, and copes very well with different sampling densities or noise levels present in the neighborhood. A Point Feature Histogram representation is based on the relationship among the points in the $k$-neighborhood and their estimated surface normals.

Due to the high computational complexity of PFH, a simplification called FPFH [Rus09] is proposed. In FPFH, the feature computing is just performed between the query point and its neighbors, rather than each pair of points within its neighborhood. FPFH is used as a local shape feature in the segmentation task in [PASW13].

**Spin Image:** The spin image is a surface representation technique that was introduced in [Joh97]. Spin images encode the global properties of any surface in an object-oriented coordinate system rather than in a viewer-oriented coordinate system. By using object-oriented coordinate systems, the description of a surface or an object is view-independent and it does not change as the viewpoint changes. Given a 3D point and its normal, it represents the distribution of the projections of all points on the point cloud to the tangent plane of the query point. In [MKRVG15], spin image is used as a feature for a semantic segmentation task.

### Training Techniques

In a semantic segmentation task, a classifier is usually trained based on the extracted features in order to obtain semantic labels for each pixel in the test phase.

**Support Vector Machine (SVM):** Originally, SVM [CV95] was proposed for linear two-class classification with margin, where margin means the minimal distance from the separating hyper-plane to the closest data points. SVM seeks for an optimal separating hyper-plane, where the margin is maximal. An important and unique feature of this approach is that the solution is based only on those data points, which are at the margin. These points are called support vectors. The linear SVM can be extended to a nonlinear one when the feature space is first transformed using a set of nonlinear basis functions. In this feature space, which can be very high dimensional, the data points can be separated linearly. An important advantage of

SVM is that it is not necessary to implement this transformation and to determine the separating hyper-plane in the possibly very-high dimensional feature space, instead a kernel representation can be used, where the solution is written as a weighted sum of the values of a certain kernel function evaluated at the support vectors. As an example, SVMs are used to train the classifier [YHRF12] for a semantic segmentation task.

**Random Forest:** Random Forests were first proposed in [Ho95]. This type of classifier applies techniques called ensemble learning, where multiple classifiers are trained and a combination of their hypotheses is used. In the case of Random Forests, the classifiers are decision trees. A decision tree is a tree where each inner node uses one or more features to decide in which branch to descend, and each leaf represents a class. One strength of Random Forests compared to many other classifiers like SVMs and neural networks is that the scale of measure of the features (nominal, ordinal, interval, ratio) can be arbitrary. Random Forests are applied to train the classifier in [SJC08] for segmentation.

### 2.1.2 Convolutional Neural Networks based Semantic Segmentation

Traditional semantic segmentation methods are usually restricted to the limited discriminative power of hand-crafted pixel features. To overcome the limitation of the hand-crafted features, Convolutional Neural Networks (CNNs) were proposed to learn feature extractors from a set of training data. A CNN can be interpreted as the combination of a set of feature extractors and a classifier, where we train the network to extract better features and classify them into different classes. The classification errors of the training samples are back-propagated to update the parameters of the network, while the classification error is optimized. CNNs were first applied in an image classification task [KSH12], in which a deep network with millions of parameters was trained on large scale datasets, in order to learn robust feature extractors for classifying images. As with outstanding achievements in CNNs based image classification,

CNNs have also had enormous success on segmentation problems.

One of the popular initial CNNs based approaches was patch classification where each pixel was separately classified into classes using a patch of image around it. The main reason to use patches was that classification networks usually have fully connected layers and therefore required fixed size images.

In 2015, Fully Convolutional Network (FCN) [LSD15] by Long et al. popularized CNN architectures for dense predictions without any fully connected layers. This allowed segmentation maps to be generated for images of any size and was also much faster compared to the patch classification approach. Almost all the subsequent state-of-the-art approaches on semantic segmentation adopted this paradigm.

Apart from fully connected layers, one of the main problems of using CNNs for segmentation are the pooling layers. Pooling layers increase the field of view and are able to aggregate the context while discarding the "where" information. However, semantic segmentation requires the exact alignment of class maps and thus, needs the 'where' information to be preserved. Two different classes of architectures evolved in the literature to tackle this issue.

The first one is the encoder-decoder architecture. Encoder gradually reduces the spatial dimension with pooling layers and decoder gradually recovers the object details and spatial dimension. There are usually shortcut connections from encoder to decoder to help the decoder recover the object details better. U-Net [RFB15] is a popular architecture from this class. It consists of a contracting path to capture context in the encoder and a symmetric expanding path from the encoder layers to the decoder layers that enables precise localization. Seg-Net [BKC17] is proposed based on FCN. It introduces more shortcut connections between encoder and decoder. Furthermore, it copies the indices from the max-pooling layers in the encoder to the decoder instead of copying the encoder features as in FCN, which makes easier for SegNet to recover the spatial information and is more memory efficient than FCN.

Architectures in the second class use what are called dilated/atrous convolutions [YK16, CPK$^+$16]. Pooling layers help in classification networks because they increase the receptive field of a network. But, as mentioned, this is not suitable for a seman-

tic segmentation task, since pooling drops the spatial information and decreases the resolution. Atrous/Dilated convolutions can compute responses at all image positions with an $n$ times larger receptive field if the full resolution image is convolved with a filter 'with holes', in which the original filter is upsampled by a factor $n$, and zeros are introduced in between filter values. Although the effective filter size increases, it is only necessary to take into account the non-zero filter values, hence both the number of filter parameters and the number of operations per position stay constant.

**CNNs based Semantic Segmentation using RGB-D data**

CNNs based segmentation approaches require a large mount of annotated data, which makes its application to RGB-D data difficult, due to the lack of annotations in RGB-D datasets. Nevertheless, there are approaches that directly encode the depth map as an extra input for training the networks. In this case, CNNs may benefit from the richer information input. For instance, [GGAM14] proposes to incorporate the depth map encoded as a pixel feature map in the training process of a CNN architecture. However, data augmentation, such as flipping, cropping etc, plays an important role in these approaches.

Another way to take advantage of the depth information in a CNNs based semantic segmentation approach is to follow a multi-task learning scheme, in which depth estimation and semantic segmentation are tackled in a hybrid CNN architecture. Since depth estimation and semantic segmentation are two strongly correlated tasks, addressing them into unified approaches can be mutually beneficial. In this case, the feature extractors in the hybrid network are better trained to cope with both semantic segmentation and depth estimation, which implicitly incorporates the depth information in CNNs.

These hybrid networks are usually designed as a combination of networks working for a single task. It is worthwhile to first study representative single task based approaches before reviewing approaches under multi-task learning schemes. In Section 2.1.2, we have studied the state-of-the-art CNNs based semantic segmentation approaches. In the following paragraphs, we first review CNNs based depth estimation

approaches, then address the multi-task based approaches in the state-of-the-art.

For the task of CNNs based depth estimation from monocular images, one of the first efforts was made by Eigen et al in [EPF14]. This approach estimates a low resolution depth map from an input image as a first step, then finer details are incorporated by a fine-scale network that locally refines the low resolution depth map using the input image as a reference. Additionally, the authors introduced a scale-invariant error function that measures depth relations instead of scale. Ivanecký [Iva16] presents an approach inspired in [EPF14], incorporating estimated gradient information to improve the fine tuning stage. Additionally, this work applies a normalized loss function leading to an improvement in depth estimation.

On the other hand, approaches addressing depth estimation and semantic segmentation with multi-task learning schemes currently receive large attention due to its potential of improving the performance in multiple tasks. In [WSL+15] a unified framework was proposed that incorporates global and local prediction under an architecture that learns the consistency between depth and semantic segmentation through a joint training process. Another unified framework is presented in [EF15] where depth map, surface normals and semantic labeling are estimated. The results obtained by [EF15] outperformed the ones presented in [EPF14] proving how the integration of multiple tasks into a common framework may lead to a better performance of the tasks. A more recent multi-task approach is introduced in [MPK16]. The methodology proposed in this work makes initial estimations for depth and semantic label at a pixel level through a joint network. Later, depth estimation is used to solve possible confusions between similar semantic categories and thus to obtain the final semantic segmentation. Another multi-task approach by Teichmann et al. [TWZ+16] presents a network architecture named MultiNet that can perform classification, semantic segmentation and detection simultaneously. They incorporate these three tasks into a unified encoder-decoder network where the encoder stage is shared among all tasks and specific decoders for each task producing outputs in real-time. This work efforts were focused on improving the computational efficiency for real-time applications as autonomous driving. A similar approach is Pixel Level Enconding and Depth Layer-

ing (PLEDL) [UCFB16], which extended FCN [LSD15] with three output channels jointly trained to obtain pixel-level semantic labeling, instance-level segmentation and 3D depth estimation.

## 2.2 Unsupervised Segmentation

Different from semantic segmentation approaches, where a group of semantics can be predefined before solving the segmentation problem, unsupervised segmentation approaches aim at tackling image segmentation problems in more general scenarios. While semantic segmentation approaches store information about the semantics they were trained to segment under the supervision of training data, unsupervised segmentation approaches try to detect consistent regions or region boundaries with respect to generic features. Although unsupervised segmentation approaches can hardly be semantic, they still provide generic representations of an image, which can be used in supervised segmentation as another source of information or to refine a segmentation when more information, such as temporal information in videos, is available. Since unsupervised segmentation problems are well-studied in recent decades, a large amount of approaches were proposed. We review some of the most representative unsupervised approaches as follows:

### 2.2.1 Clustering Algorithms

Clustering algorithms can directly be applied on the pixels, when one gives a feature vector per pixel. Two famous clustering algorithms are $k$-means and the mean-shift algorithm. The $k$-means algorithm is a general-purpose clustering algorithm which requires the number of clusters to be given beforehand. Initially, it places the $k$ centroids randomly in the feature space. Then it assigns each data point to the nearest centroid, moves the centroid to the center of the cluster and continues the process until a stopping criterion is reached. A faster variant is described in [HH75]. k-means was applied by [CLP98] for medical image segmentation.

Another clustering algorithm is the mean-shift algorithm which was introduced by [CM02] for segmentation tasks. The algorithm finds the cluster centers in a feature space by initializing centroids at random seed points and iteratively shifting them to the mean position in the feature space within a certain range. Instead of taking a hard range constraint, the mean can also be calculated by using any kernel. This effectively applies a weight to the coordinates of the points. The mean shift algorithm finds cluster centers at positions with a highest local density of points.

### 2.2.2 Watershed Segmentation

The watershed algorithm takes a feature, such as the gradient magnitude, from a grayscale image and interprets it as a height map. Low values are catchment basins and the higher values between two neighboring catchment basins form the watershed lines. In order to detect the homogeneous regions of an image, the watershed is usually applied on a variational feature on the image. The algorithm starts to fill the basins from the lowest point. When two basins are connected, a watershed is found. The algorithm stops when the highest point is reached. A detailed description of the watershed segmentation algorithm is given in [RM00]. An as example, the watershed segmentation was used in [JLD03] to segment white blood cells. As the authors describe, the segmentation by watershed transform has two flaws: Over-segmentation due to local minima and thick watersheds due to plateaus.

### 2.2.3 Active Contour Models

Active contour models (ACMs) are algorithms that segment images roughly along edges, but also try to find a border which is smooth. This is done by the minimization of an energy function computed on the resulting contour. They were initially described in [KWT88]. ACMs can be used to segment an image or to refine segmentation as it was done in [AM98] for brain MR images.

### 2.2.4 Graph based Segmentation

Graph-based image segmentation algorithms typically interpret pixels as vertices and an edge weight is a measure of dissimilarity, such as the difference in color [FH04]. In this case, there are several different candidates for edges, such as the 4-neighborhood or an 8-neighborhood.

In graph based approaches, a segmentation is a partition of the vertex set into segments, where each segment corresponds to a connected component in the graph. Graph based approaches obtain a segmentation by removing edges connecting vertices, while producing non-connected sub-graphs (segments). One intuitive way to cut the edges is by first building a minimum spanning tree of the graph, then removing edges on the minimum spanning tree above a threshold [Zah71] to obtain the segments. This threshold can either be constant, adapted to the graph or adjusted by the user. After the edge-cutting step, the connected components are the segments. However, since differences between pixels within high variability regions can be larger than those between the ramp and the constant regions, it is difficult to find an adequate threshold. In [FH04], the authors propose a more efficient way to approach the graph based image segmentation problem, in which pixels are iteratively merged into segments by comparing the internal difference of a segment (measured by the maximum edge weight within the minimum spanning tree of the segment) and the difference between segments (represented by the minimum edge weight connecting the two segments).

Another way to perform graph based image segmentation is based on cutting the edges with minimum weights in a graph, where the cut criterion is designed to minimize the similarity between vertices on the graph that are being split. This problem is solved by graph cut algorithms, such as Stoer and Wanger algorithm [SW97], Ford-Fulkerson algorithm [FF56], etc. The work in [WL93] introduces such a cut criterion for image segmentation purposes. However, it is biased towards finding small segments. In [SM00], the bias is addressed with the normalized cut criterion, which takes into account both the total dissimilarity between the different segments as well as the total similarity within each segment.

The large amount of vertices and edges in the graph constructed directly from

pixels leads to a large scale problem of graph based image segmentation. Additionally, the graph constructed directly from pixels is also very sensitive to the pixel noise. In this case, a prior pixel grouping step is usually introduced to abstract pixels into locally homogeneous patches, called super-pixels. For instance, the work in [ASS+12] proposes a Simple Linear Iterative Clustering (SLIC) method, which adapts a k-means clustering approach to efficiently generate super-pixels. In [CM02], mean shift is applied to find modes (super-pixels) in a color or intensity feature space. The adjacency graph of the obtained super-pixels is usually employed in graph based image segmentation approaches, in order to reduce the number of vertices and edges in the graph while preserving important boundary information.

### 2.2.5 RGB-D based Unsupervised Segmentation

Similar to RGB-D based supervised approaches, RGB-D based unsupervised approaches extended from RGB methods, and usually add depth values as an extra feature of the data, such as [WGB12]. Apart from that, there are also methods working with 3D point clouds instead of depth values. For instance, [CSSPW14] begins by decomposing the 3D point cloud of the scene into an adjacency-graph of surface patches based on a voxel grid. Edges in the graph are then classified as either convex or concave using a novel combination of simple criteria which operate on the local geometry of these patches. In this way, the graph is divided into locally convex connected sub-graphs, which with high accuracy represent object parts. In [PASW13], Papon et al. propose a novel unsupervised over-segmentation approach that uses voxel relationships to produce over-segmentations, which are fully consistent with the spatial geometry of the scene in three dimensional, rather than projective space. Enforcing the constraint that segmented regions must have 3D spatial connectivity prevents super-voxels from flowing across object boundaries.

## 2.3 Video Segmentation

Video segmentation methods usually extend image segmentation to video frames by considering the temporal coherence of object segments. Approaches in the state-of-the-art can be broadly classified considering two aspects: the input source and the segmentation purpose.

The input source splits the approaches into two categories: 1) frame-by-frame based approaches, where one or two successive frames of a video are provided as input, 2) volume-based approaches, where many successive frames of a video are input at once.

Frame by frame approaches have the advantages of low memory requirement and being able to directly cope with a video stream online, while volume-based approaches benefit from the richer input information, so that long term temporal correspondences can be employed for maintaining segmentation coherence.

The segmentation purpose distinguishes the approaches into two categories: 1) video foreground objects extraction and 2) video frame partitioning.

### 2.3.1 Video Foreground Object Extraction

Approaches addressing video foreground object extraction usually focus on specific types of foreground objects, where semantic image segmentation methods can be smoothly extended. In this case, training samples for specific objects can be easily collected. Object models are learned from training samples, in order to segment foreground objects frame by frame in a sequence. For instance, the work in [CMPT+17] trains an appearance model of a foreground object based on CNNs to segment objects in each frame. More generally, in [PKB+17], the authors train the CNNs to predict generic foreground object segmentation with respect to the segmentation in the previous frame. In [SYRK+17], CNNs are trained to perform pixel-level matching, so that the annotated foreground object can be tracked and segmented in the video.

Approaches employing generic models for objects in scene also relax the constraint of training specific object models. They learn incrementally and update the object

models frame by frame. This idea was first explored in object tracking approaches, where the target objects are provided as initialization for generic models. From the perspective of video foreground object segmentation, an initialization is usually not available. Thus, the generic models are corresponded and updated in each frame. For instance, in [HDT15], the temporal correspondences are built between 3D surfaces which ideally represent the point cloud of objects. The strategy of building temporal correspondences is simply defined based on the overlapping size of surfaces in consecutive frames. The author reports the drawback of their strategy of building temporal correspondences in [HDT15] indicating that fast movements are not well handled in the approach.

Several methods represent the raw data with a pool of object-like regions [LKG11, ML12, ZJS13]. These object proposals are extracted from each frame based on generic spatial features. Then, temporal relations and motion are used to extract the primary object segment in the video sequence by means of optimization techniques. In [ZJS13], for instance, the selection is formulated as the longest path problem for a Directed Acyclic Graph. The work in [FXLL15] uses this approach for co-segmentation, introducing in this case RGB-D images. In this line, and working with RGB-D videos, the work in [FXL17] proposes to select the significant objects from a pool of object proposals through graph optimization, introducing objectness, motion and RGB-D video saliency to evaluate the importance of each object proposal. These approaches are generally computationally expensive, mainly due to the cost of the proposal generation process, and can not handle a varying number of objects in the sequence.

### 2.3.2 Video Frame Partitioning

On the other hand, approaches tackling frame partitioning in a video usually focus on obtaining a partition for each frame in a video, which makes it hard to predefine the types of objects in the back ground. In this case, more generic unsupervised image segmentation approaches are usually employed. Unsupervised approaches can hardly achieve object level segmentation due to the lack of information. However, locally ho-

mogeneous segments obtained in those approaches form a good frame representation for the temporal analysis in a video. Analyzing temporal information helps unsupervised approaches so that segments in a single frame can be grouped with respect to the temporal coherence, which may produce object level segmentation. Hierarchical representation is the most representative approach. Hierarchical representations are widely employed in methods, such as [AMFM11, XCGN17, PS13, HBEC14], to represent the raw data from coarse to fine. A hierarchical representation usually starts from the segments at a relatively fine level, such as super-pixels or over-segmented regions recovered from a contour probability map [AMFM11] for RGB data, or super-voxels [PKAW13, CSSPW14] in the case of RGB-D data. Then, the segments are gradually grouped into coarser level regions following strategies like Binary Partition Tree (BPT) [VMS08], Multiscale Combinatorial Grouping [APTB$^+$14] or Shape-Space Filtering [XCGN17]. One advantage of hierarchical representations in video segmentation is that temporal correspondences can be established at different scales, which benefits the temporal coherence for the video segmentation task considered afterwards. On the other hand, exploiting temporal coherence also helps to better construct the hierarchical representation at each frame. For instance, in [PS13], long term trajectories are leveraged to help building BPTs.

### 2.3.3 Building Temporal Correspondences

An important point in different video segmentation tasks is how to leverage the temporal information. In this section, we review how temporal coherence is introduced in different video segmentation methods.

In hierarchical representation based approaches, such as [GKHE10], temporal correspondences are built between segments at an over-segmentation level. In this case, temporal neighborhood is simply used as the criterion to build the temporal correspondences in successive frames, which might not be a suitable strategy for scenes with fast moving objects. The authors also propose to build the temporal correspondence based on optical flow. But the global optimum in the correspondence building

task is still difficult to achieve considering the large scale of the problem. Instead, a local optimum is usually accepted as a solution for the correspondence problem, which makes the established correspondences less reliable in the subsequent analysis. Similarly, in [HBEC14] the authors propose to build the temporal correspondence in 3D space using depth information, which helps building more accurate temporal correspondences between over-segments along the sequence.

Instead of building temporal correspondences at such a fine level, approaches employing generic models for objects in scene build temporal correspondences at object level. For instance, in [HDT15], the temporal correspondences are built between 3D surfaces which ideally represent the point cloud of objects. The strategy of building temporal correspondences is simply defined based on the overlapping size of surfaces in consecutive frames. The author reports the drawback of their strategy of building temporal correspondences, that fast movements are not well handled in the approach.

In object proposal based approaches, such as [ZJS13], the temporal correspondence building task is converted into an optimization problem on the graph, in which an optimal path is obtained among object proposals in video frames. The obtained path along time shows the temporal correspondence.

## 2.4 Summary

In this chapter, we have reviewed segmentation approaches in the state-of-the-art solving different segmentation problems.

Specifically, we have compared traditional semantic segmentation methods with CNNs based approaches for both RGB and RGB-D data, which shows the trend of exploiting CNNs to learn adequate feature extractors from annotated data, instead of hand-crafting features in the traditional methods. The main challenge of applying semantic segmentation to RGB-D data is the lack of annotations in RGB-D datasets. To cope with this, two research lines are studied, which are data augmentation and multi-task learning schemes.

Apart from the semantic segmentation problem, we have also reviewed both RGB

and RGB-D based unsupervised image segmentation approaches which could serve as a building block for more powerful supervised methods in the future, and naturally have the advantage of dealing with generic scenes. The state-of-the-art strategies of incorporating depth information in unsupervised methods are studied.

However, unsupervised image segmentation approaches can hardly obtain higher level segmentation (such as object segmentation) due to the lack of prior knowledge. In this case, temporal information contained in video sequences may help better defining the segmentation criterion so that higher level segmentation could be obtained. We have reviewed RGB and RBGD based video segmentation approaches, and especially the strategies of building and exploiting temporal correspondences between consecutive video frames.

In the following a few chapters, we start to explain our work which contributes to several segmentation problems with respect to the current background that we have introduced in Chapter 1 and 2.

Chapter 3

# Semantic Segmentation based on RGB-D data

## 3.1 Introduction

Prior work in semantic segmentation includes many different approaches that relied on hand-crafted features combined with flat classifiers such as Support Vector Machines [CV95] or Random Forests [Ho95]. However, the performance of these methods has always been limited by the discriminative power of the features. More recently, due to the successful application of Convolutional Neural Networks (CNNs) in image classification, semantic segmentation, treated as a dense classification task, is also addressed in some CNNs based approaches. These approaches leverage the outstanding representation power of CNNs by training CNNs to extract robust features rather than exploiting manually crafted features.

On the other hand, depth information captured by a consumer depth sensor, serving as an extra source of input data with geometric information, also widely helps improving approaches in different computer vision tasks, such as image segmentation, object recognition, etc. Thus, exploiting depth information in CNNs based methods becomes a promising way to better approach the semantic segmentation problem. However, CNNs based methods usually require a large amount of annotated data for training the network, which is not sufficiently available in RGB-D datasets. Besides,

the annotated RGB-D datasets are usually restricted to indoor scenes due to the technical limitation of consumer depth sensor, such as limited range for depth sensing and poor robustness to ambient infrared noise in outdoor scenes. The lack of annotated data and the restricted scenes involved in RGB-D dataset limit the approaches that directly train a neural network for extracting features by employing depth maps as another channel of the input.

Similar to semantic segmentation, depth estimation is also an important task in image understanding that helps to comprehend the structure of the environment depicted in an image and the objects on it. Due to the strong correlation between the depth estimation and semantic segmentation, approaching these two tasks together may be mutually beneficial. In this situation, depth maps are used as one type of ground truths for defining the target of a multi-task learning process, which provides a promising way to implicitly exploit depth information in a semantic segmentation task.

In the following sections, we present our methodology which jointly tackles image segmentation and depth estimation using RGB-D data via a multi-task learning framework in CNNs.

## 3.2 Our Proposal

Depth estimation and semantic segmentation are two widely studied problems in the image processing community and recently have been tackled through deep learning techniques due to its successful results in terms of accuracy and efficiency. In Chapter 2, we have reviewed the related work in the state-of-the-art, studying different CNN based semantic segmentation methods and depth estimation methods which work for a single task (either semantic segmentation or depth estimation), as well as the state-of-the-art methods jointly approaching semantic segmentation and depth estimation under the multi-task learning schemes.

In this section, we introduce our approach called HybridNet, which also follows the multi-task learning schemes. More precisely, our approach aims to directly esti-

mate the segmentation and depth maps from an input color image by unifying CNNs working for a single task into a sole hybrid convolutional neural network. The idea of merging two tasks in one architecture is motivated by the fact that both depth and segmentation maps represent strong geometrical information of a scene. Multi-task approaches in the state-of-the-art seek to extract features suitable to perform diverse tasks at a time, which lead to an improvement in both estimated information and simplification of systems where multiple modalities are required, such as autonomous navigation [SKK16], robotics [BRE+17] or augmented reality [AM17].

Most of the state-of-the-art works unify tasks under a feature extraction block whose output becomes the input of a group of decoders designed to carry out each task. In our approach, we investigate the common attributes as well as the distinction for depth estimation and semantic segmentation and clarify how the two tasks help with each other in a hybrid system. We propose to use a global depth estimation network to estimate separately the global layout of a scene from the input image additionally to the common features extraction. The main motivation to incorporate this extra step is based on the idea that the features network will focus better on extracting common features working for both tasks by separating the global information extraction only needed in the depth estimation task during the training process. The modularized features extraction process helps on producing better features, which leads to an improved refined depth map and segmentation.

The rest of the chapter is organized as follows: in Section 3.3 we introduce the proposed methodology, the detailed explanation of the proposed architectures are presented in Section 3.4, as well as the training details. In Section 3.5, we present the experiment results of our approach in different datasets and compare our approach with state-of-the-art approaches. Finally, conclusions are drawn in Section 3.6.

## 3.3 Hybrid Convolutional Framework

In this section a general explanation of our hybrid convolutional model and its application to depth estimation and semantic segmentation is presented. To this end,

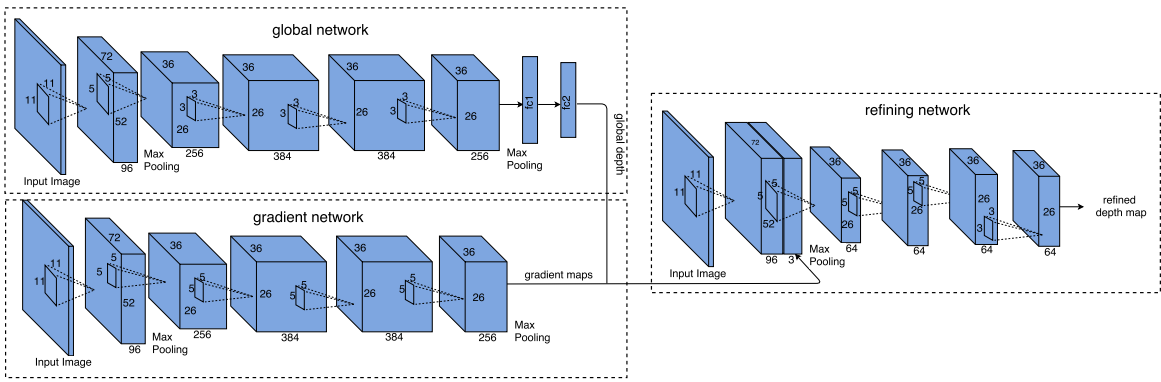



Figure 3.1: Depth estimation network.

a description about the two single task architectures [Iva16, CPK+16] employed in our approach is first presented. Then, we describe the proposed hybrid model along with a discussion to approach the problem of how to unify two tasks under one sole framework.

The depth estimation architecture [Iva16], denoted as DepthNet in this thesis, is made of three components: global depth network, gradient network and refining network, as shown in Fig.3.1. These three components all follow AlexNet structure. DepthNet first estimates a depth map of the scene at a global level from the single input RGB image via a global depth network. Meanwhile, it predicts two depth gradient maps from the input RGB image via a gradient network. Finally, a refining network uses the input image along with depth gradient maps to locally refine the global depth map and thus produce a better detailed depth map. As explained in [Iva16], the three components in DepthNet are trained separately. For training the global depth network, the downsampled depth maps are used as the ground truth. Beside the global depth network, the gradient network is trained based on the magnitude of depth gradient on $x$ and $y$ direction computed from the depth map. Along with the global depth network and gradient network, the refining network is again trained on the downsampled depth maps in the training data.

There are two main reasons to consider employing DepthNet as the depth estimation component in our approach: 1) DepthNet follows the state-of-the-art framework for depth estimation which is representative for a brunch of methods. 2) DepthNet

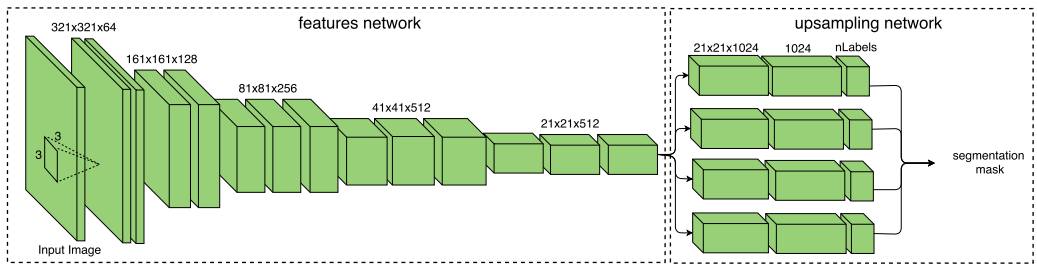



Figure 3.2: Semantic segmentation network.

has a modularized architecture, which allows us to analyze each of the components in it and better integrate DepthNet into a hybrid architecture.

The semantic segmentation architecture [CPK+16] as shown in Fig.3.2, is divided in two main parts: features network and atrous up-sampling network. For the features network, it follows the VGGNet architecture proposed in [SZ14]. It is in charge of extracting robust features from the input image, which benefits from the deep structure of the network. On the other hand, the atrous up-sampling network is a group of atrous spatial pyramid pooling layers [CPK+16] (see Chapter 2, Section 2.1.2 for detailed explanation about the atrous pooling layer) which outputs a class score map with the number of channels equal to the number of labels. Atrous upsampling layers allows us to explicitly control the resolution at which feature responses are computed within the architecture, while enlarging the field of view of filters in order to incorporate larger context in the semantic segmentation task. The semantic segmentation architecture is denoted as Deeplab-Atrous Spatial Pyramid Pooling (DeepLab-ASPP) in this thesis. In DeepLab-ASPP, all parts are trained together.

DeepLab-ASPP is employed as the semantic segmentation component in our approach due to its outstanding performance in this task.

### 3.3.1 Unifying Single Task Architecture for Multi-Tasks

Considering the functionality of each component in DepthNet and DeepLab-ASPP, we propose and compare two different hybrid architectures for the joint depth estimation and semantic segmentation task.

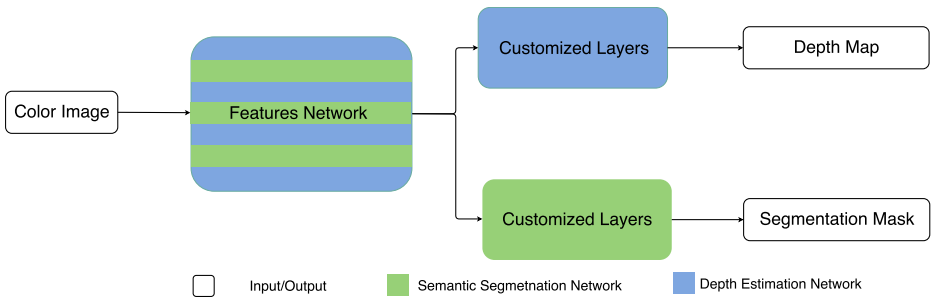



Figure 3.3: Architecture 1

**Architecture 1**: An intuitive way to unify two tasks in a sole architecture is to totally share the feature extraction process for both tasks. It follows the idea from the most representative architectures in the state-of-the-art [EF15, MPK16, TWZ+16], in which a common convolutional network is shared for extracting features. Following the feature extraction block, customized layers are used for each task, in order to decode the commonly extracted features and apply them in different tasks. Sharing the feature extraction process for different tasks with a common convolutional network links the two tasks, since the parameters of the shared network are optimized with respect to the losses defined on both tasks in the training phase. The advantage of this architecture is obvious. Since most of the layers are shared for both tasks, less parameters are involved in the training process, which makes it easy to be trained. In practice, we exploit the VGG structure [SZ14] as the features extraction network for both tasks. Based on the extracted features, the atrous up-sampling network in DeepLab-ASPP is employed for the semantic segmentation task, while the refining network in DepthNet is leveraged as the decoder for the depth estimation task. We denote the architecture 1 as HybridNet A1 in the rest of this chapter.

**Architecture 2**: The idea of building this architecture is to further clarify the common and specific attributes in the two tasks. To this end, we employ DepthNet for depth estimation, in which the architecture is better modularized as global depth network, a gradient network and a refining network, and DeepLab-ASPP for semantic segmentation. We propose to unify the two networks into a hybrid network by substituting the gradient network of DepthNet to the features network of DeepLab-ASPP.

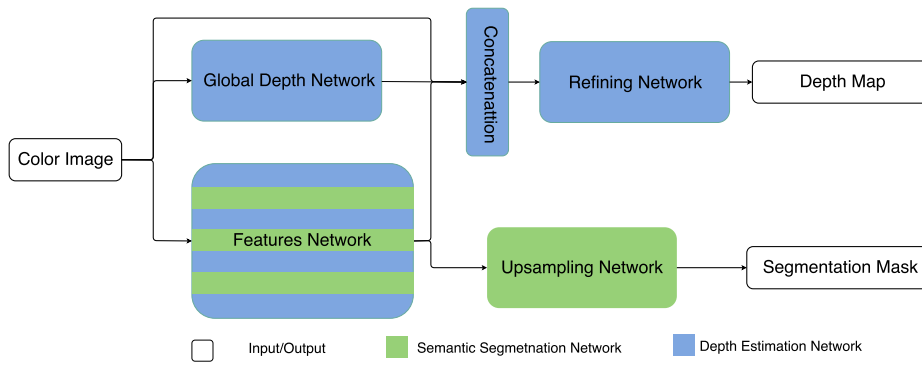



Figure 3.4: Architecture 2

In this manner, the features network is shared by both networks. The advantages of this hybrid architecture are two folds. On one hand, the strong power of extracting object information from a color image learned in the semantic segmentation task can also benefit depth layering when predicting a depth map, while the strong power of extracting rich depth boundaries from a color image learned in the depth estimation task is shared in the semantic segmentation task to improve the segmentation accuracy on object boundaries. On the other hand, the global layout of the scene which is more concerned in depth estimation than semantic segmentation is estimated independently by a global network in the depth estimation task, which avoids interfering the common feature extraction for both tasks. In practice, we keep the global network and refining network in DepthNet the same, while replacing the structure of gradient network with the VGG structure in order to keep the structure consistent with the features network in DeepLab-ASPP. We denote the architecture 2 as HybridNet A2 in the rest of this chapter.

## 3.4 Architecture Details

Since the proposed architectures are assembled with basic components in the two single task architectures, we explain the detail of the proposed architectures by describing the two single task architectures in this section.

### 3.4.1 Depth Estimation Network

As described in Section 3.3, depth estimation network is modularized to calculate a refined depth map from a single input image through a three stages convolutional network. As shown in Fig.3.1, global depth network is formed by 5 convolutional layers and two fully connected layers, which corresponds to the architecture of AlexNet [KSH12]. Following each convolutional layer, a Rectified Learnear Unit (ReLU) is introduced as an activation function to provide non-linearity to the system. Local normalization is also performed after each convolutional layer in a Local Response Normalization layer (LRN), which helps the generalization of the system. The local normalization is done by dividing each input value by $(1 + (\alpha/n \sum_i x_i^2)^\beta)$, where $n$ is the size of the local region where the sum is being evaluated over. Max pooling layers are placed after the first and the last convolutional layer to provide basic translation invariance to the internal representation and reduce the number of parameters of the system. In this network max-pooling is performed over a $3 \times 3$ window with a stride of 2. Since the global depth network aims at describing the global layout of the scene, we introduce two fully connected layers following the last convolutional layer, in order to capture the information contained in the intermediate representation with the full receptive field. In practice, 1024 neurons are included in the first fully connected layer while 1681 neurons are included in the second fully connected layer. We reshape the 1681 neurons to a 41 by 41 matrix which is treated as the output of the global depth network. In this manner, we predict a global depth map with $\frac{1}{8}$ resolution for an input image with size 321 by 321.

The gradient network aims at estimating depth gradient from an input color image. In practice, we employ the same architecture used in global depth estimation except for the fully connected layers.

Finally, refining network takes the concatenation of the output from global depth network, gradient network and the color image as input and compute a refined depth map. Refining network improves the rough estimate from the global depth network, utilizing gradients estimated by the gradient network and an input color image. In practice, first convolutional layer processes the input color image, followed by a ReLU

layer, a LRN layer and a max pooling layer, which produces the feature maps extracted from the color image. These feature maps are concatenated with outputs of the global depth network and the gradient network, then are fed to remaining four convolutional layers. Each of them is followed by a ReLU layer. The output from the 5th convolutional layer in the refining network is treated as the output (a refined depth map with size 81 by 81).

### 3.4.2 Semantic Segmentation Network

Fig.3.2 presents an overview of semantic segmentation network. In this figure, it is shown in a detailed manner how the input image is processed by first going through a group of convolutional layers for feature extraction (features network) and then through an up-sampling procedure which finally provides the segmentation map (up-sampling network). Dividing this architecture into two parts helps us to understand it as a single task network but also how it can be integrated into a hybrid model.

The features network contains 5 groups of convolutional layers, forming a deep architecture. All of these convolutional layers have the same kernel size $3 \times 3$. For simplicity, we only plot the convolutional kernel in the first convolutional layer in the features network. Following each convolutional layer, a ReLU layer is provided as the activation function. Pooling layers are placed after each group of convolutional layers to reduce the computational cost by down-sampling the internal representation, as well as to provide basic translation invariance to the internal representation.

On the other hand, atrous up-sampling network contains 4 parallel groups of three convolutional layers, in order to perform upsampling operation at different scales. Each branch upsamples the output from the features network at the first convolutional layer with an atrous convolutions. An atrous convolution employs a dilated convolution template, in which a convolution template is enlarged by filling zeros with respect to a defined rate. In this manner, we can explicitly control the resolution of upsampling operation and enlarge the field of view of filters to incorporate larger context in the semantic segmentation task without introducing more parameters. In

practice, we employ atrous convolutions with rates $6, 12, 18, 24$ respectively for each branch. The other 2 convolutional layers in each branch perform $1 \times 1$ convolutions, which increases the non-linearity of the decision function without affecting the receptive fields of the convolutional layers. Taking the output of the 4 branches of up-sampling layers as input, a soft-max layers produces the final semantic segmentation mask.

### 3.4.3 Training Details

As explained in the Section 3.3.1, the two proposed architectures (HybridNet A1 and A2) are based on DeepLab-ASPP [CPK+16] and DepthNet [Iva16]. Although HybridNet A1 and A2 are constructed by merging single task architectures, the training process for the hybrid architectures are not always performed the same like in those single task architectures.

In HybridNet A1, we initialize the features network and the atrous up-sampling network with the model provided by DeepLab [CPK+16] which was pre-trained for classification purpose on ImageNet. The rest parts in HybridNet A1 are initialized using a Random Number Generator (RNG). The RNG is commonly set to be a Gaussian distribution with zero mean and 0.01 variance.

In HybridNet A2, we initialize the features network and upsampling network before the training process using again the model provided by DeepLab [CPK+16]. Additionally, we initialize the global depth network using the model provided in [Iva16]. The rest of the other parts in HybridNet A2 are randomly initialized using a the same RNG.

Once we have the initialization for our hybrid architecture, all its components are trained simultaneously. Both of the hybrid architectures are trained for 100K iterations with a learning rate $2.5 \times 10^{-6}$, polynomial learning rate decay policy. We choose a relatively low learning rate to avoid over-fitting the model, whereas the number of iterations is set to 100K to ensure the convergence of the training process. The momentum is set to 0.9 and weight decay 0.005. These hyper-parameters are

set following the configuration proposed in [CPK+16]. The input image is randomly cropped with a size $320 \times 320$. We set batch size to 7, regarding the maximum allowance of memory.

The loss function used in both architectures is the same. For the semantic segmentation task, $L_S$ is the sum of the cross-entropy terms for each spatial position in the output class score map, being our targets the ground truth labels. All positions and labels of the output class score map are equally weighted in the overall loss function with the exception of those unlabeled pixels which are ignored. The loss function utilized for the depth estimation task is made by two Euclidean loss layers $L_{D_{abs}}$ and $L_{D_{mvn}}$. $L_{D_{abs}}$ computes Euclidean distance between absolute values of a depth map in the ground truth and the estimated depth map, while the $L_{D_{mvn}}$ computes the Euclidean distance between estimation and ground truth after performing a mean variance normalization on both of them. Then, the hybrid loss function $L_H$ is therefore defined as the linear combination of them:

$$L_H = \alpha L_S + (L_{D_{abs}} + L_{D_{mvn}}) \tag{3.1}$$

where $\alpha$ is the term used to balance the loss functions of depth estimation and semantic segmentation tasks. For training our hybrid model we defined $\alpha = 1000$.

## 3.5 Experiments

We quantify the performance of the proposed architectures on both semantic segmentation and depth estimation in different scenes using our Caffe implementation. We first evaluate the proposed architectures in road scenes which is of current practical interest for various autonomous driving related problems. Secondly, the proposed architectures are evaluated in indoor scenes which is of immediate interest to possible Augmented Reality (AR) applications.

### 3.5.1 Road Scene

In this section, we present the evaluation of the proposed architectures in road scenes. A number of road scene datasets are available for semantic parsing [BFC09, GLU12, COR+16]. Since we evaluate the proposed architecture from both semantic segmentation and depth estimation perspective, we employ Cityscapes dataset [COR+16] in our experiment, which provides not only the ground truth of semantic labels but the depth information of each frame. Cityscapes contains 5000 RGB images manually selected from 27 different cities for dense pixel-level annotation to ensure high diversity of foreground objects, background and overall scene layout. Along with each of the 5000 RGB images, Cityscapes dataset provides the depth map obtained from a stereo vision system. The 5000 images in the dataset are split by the provider into 2975 training RGB images of size $1024 \times 2048$ along with their corresponding 2D ground truth object labels for 19 outdoor scenes classes and depth information, 500 RGB images for test validation with their corresponding annotations and, for benchmarking purposes, 1525 test RGB images. In practice, the training process of our approach was performed using the 2975 images of Cityscapes training set that provides a depth map and object labels of 19 classes for each RGB image. To evaluate the performance of the proposed architectures, we group the 500 images of the validation set and the 1525 images of the test set in the Cityscapes dataset into a single evaluation set of 2025 images. In the training phase, images in the training set are shuffled and randomly cropped to fit the input image size of the hybrid architecture. Training data augmentation is done by flipping and mirroring the original images, in order to enlarge the training set. In the testing phase, we crop the test image with the original size of $1024 \times 2048$ into a group of images with the size of 321 by 321 which cover the whole test image while having the minimum overlapped area. These images are tested one by one and grouped to obtain the final prediction of the segmentation mask and depth map. For the overlapped area, we employ majority voting to determine the predicted labels on the segmentation mask. Likewise, for the overlapped

area, the predicted depth values on the depth map are computed as the mean values.

Our first aim is to determine if the features obtained in the shared part of the proposed architectures solving the two tasks simultaneously provide better results than the ones that we would obtain using two identical networks trained separately. This is why in addition to the results of the proposed architectures, we present the results obtained by the models that solve these two tasks separately for comparison. The models used to train semantic segmentation and depth estimation independently are denoted as DeepLab-ASPP [CPK+16] and DepthNet [Iva16], respectively. We trained these two models using the code provided by the authors with the same training data in Cityscapes dataset and the same training configuration than the proposed architectures. Apart from that, we also compare different ways of unifying single task architectures proposed in Section 3.3.1, in order to justify the better unifying strategy. Besides, the comparison between the proposed architectures and a hybrid method in the state-of-the-art [UCFB16] is also made in Cityscapes dataset. The hybrid approach proposed in [UCFB16] is similar to HybridNet A1, in which the encoder network in FCN [LSD15] is employed as the features network shared by three different tasks and the decoder network in FCN is then employed for each task to decode the commonly extracted features. The three tasks that [UCFB16] tackles are semantic segmentation, depth layering, boundary detection, which is similar to our target. However, in the depth layering task, [UCFB16] focuses on estimating a depth label for each object, instead of estimating the real depth value of the whole scene at pixel level. This is also the reason that we only compare the performance between our approach and [UCFB16] in semantic segmentation. We present the results in our experiments in the following two subsections specifying the evaluation in semantic segmentation and depth estimation respectively.

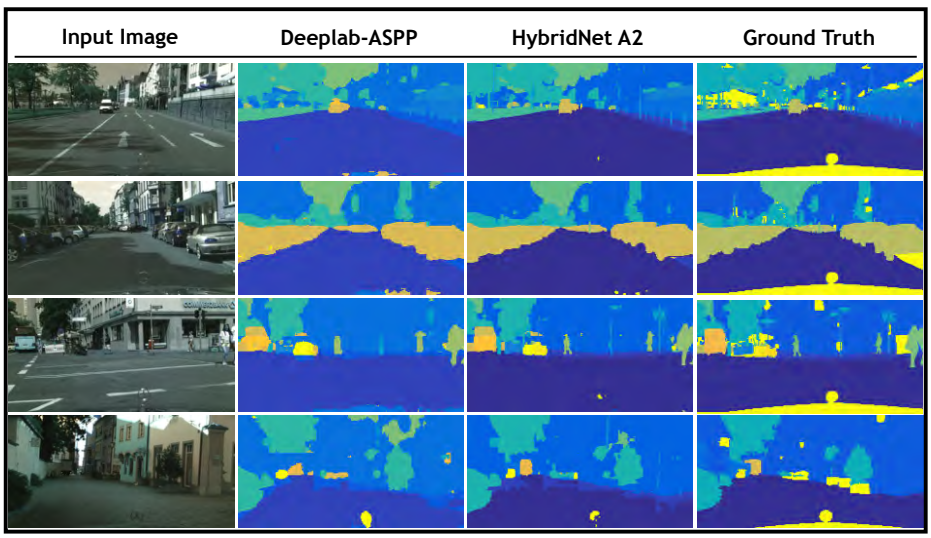



Figure 3.5: Semantic segmentation qualitative results. A comparison between semantic segmentation estimation against ground truth is presented. From left to right, input image is depicted in the first column. In column 2 the segmentation map estimated by DeepLab-ASPP semantic segmentation network [SZ14] is presented, in column 3 the estimated segmentation map by our hybrid method are presented and finally the ground truth is depicted in column 4.

### Semantic Segmentation

Fig.3.5 provides 4 examples from the evaluation set for visual comparison between the results obtained by our hybrid model and ground truth as well as those obtained by DeepLab-ASPP. The purpose of this figure is to depict the differences between a single task and a multi-task approach. In Fig.3.5 the input image is displayed in the first column, second and third columns show the results obtained by DeepLab-ASPP and our hybrid model respectively. Finally, in the fourth column the ground truth is presented for reference. This figure shows how the segmentation performed by the proposed HybridNet A2 retains with a greater detail the geometrical characteristics of the objects contained in the scene. For instance, in the 3rd row where the shapes of a pedestrian and a car can be better distinguished in the estimation obtained by Hybrid A2 than the one obtained by DeepLab-ASPP.

| | G | C | mIoU |
|---|---|---|---|
| HybridNet A2 | **93.26** | **79.47** | **66.61** |
| HybridNet A1 | 89.31 | 77.22 | 58.1 |
| PLEDL [UCFB16] | - | - | 64.3 |
| DeepLab-ASPP [CPK$^+$16] | 90.99 | 74.88 | 58.02 |
| FCN [LSD15] | - | - | 65.3 |
| SegNet[BKC17] | - | - | 57.0 |
| GoogLeNetFCN[SLJ$^+$15] | - | - | 63.0 |

Table 3.1: Evaluation of HybridNet against Multi-task and single task approaches.

In addition to qualitative results, we employ three commonly used measures, in order to measure quantitatively the segmentation performance: the Global Accuracy (G), the Class Average Accuracy (C) and mean Intersection over Union (mIoU). The global accuracy counts the percentage of pixels which are correctly labeled with respect to the ground truth labelling. The class average accuracy is the mean of the pixel accuracy in each class. The mean intersection over union measures the average Jaccard scores over all classes. Table 3.1 presents the quantitative results and confirms that the proposed HybridNet outperforms the results obtained by DeepLab-ASPP. The improvements obtained by our method against DeepLab-ASPP confirm the hypothesis that sharing the feature extraction network between tasks leads to an improvement in terms of segmentation accuracy. The strategy of unifying two single task architectures affects the segmentation performance of hybrid methods. HybridNet A2 where common and specific attributes between two different tasks are better clarified outperforms HybridNet A1 in which the feature extraction process are totally shared for the two tasks. The improvement that HybridNet A1 obtains against DeepLab-ASPP is very limited (HybridNet A1 58.1% mIoU against DeepLab-ASPP 58.02% mIoU), however, Hybrid A2 improves the mean IoU by around 8%. We also compare our architectures against a state-of-the-art hybrid method [UCFB16] in Table 3.1. HybridNet A2 has a better segmentation performance in all three measures than the work in [UCFB16]. For additional evaluation, comparisons between our approach against other well adopted single

| Metrics | Definition |
|---|---|
| PP | $max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) = \gamma < threshold$ |
| PP-MVN | $max\left(\frac{MVN(d_i)}{MVN(d_i^*)}, \frac{MVN(d_i^*)}{MVN(d_i)}\right) = \gamma < threshold$ |
| ARD | $\frac{1}{N}\sum \left\|d_i - d_i^*\right\| / d_i^*$ |
| SRD | $\frac{1}{N}\sum \left\|d_i - d_i^*\right\|^2 / d_i^*$ |
| RMSE-linear | $\sqrt{\frac{1}{N}\sum \left\|d_i - d_i^*\right\|^2}$ |
| RMSE-log | $\sqrt{\frac{1}{N}\sum \left\|log\left(d_i\right) - log\left(d_i^*\right)\right\|^2}$ |
| SIE | $\frac{1}{2N}\sum_i \left(log\left(d_i\right) - log\left(d_i^*\right) + \frac{1}{N}\sum_j \left(log\left(d_j\right) - log\left(d_j^*\right)\right)\right)^2$ |

Table 3.2: Definition of the evaluation measures for depth estimation: PP, PP-MVN ARD, SRD, RMSE-linear, RMSE-log and SIE.

task methods [LSD15, BKC17, PCMY15, CPK+16] are presented in Table 3.1.

### Depth Estimation

For depth estimation evaluation, in Fig.3.6 we present a visual comparison of the results obtained by Hybrid A2 as well as those obtained by the single task approach presented in [Iva16] against the ground truth. The figure displays, row-wise the same 4 examples depicted in Fig.3.5. Fig.3.6 depicts the input image in the first column, the depth map obtained by DepthNet in the second column, while third and fourth columns show the depth map obtained by HybridNet A2 and ground truth respectively. Note how the results obtained by Hybrid A2 are more consistent with the ground truth than those obtained by DepthNet in terms of the depth layering.

Additionally to qualitative analysis, we evaluate the performance of our methodology for depth estimation employing 6 commonly used measures: Percentage of Pixel (PP), Mean Variance Normalized Pixel of Percentage (PP-MVN), Absolute Relative Difference (ARD), Square Relative Difference (SRD), Linear Root Mean Square Error (RMSE-linear), Log Root Mean Square Error (RMSE-log) and Scale Invariant Error (SIE). Table 3.2 shows the

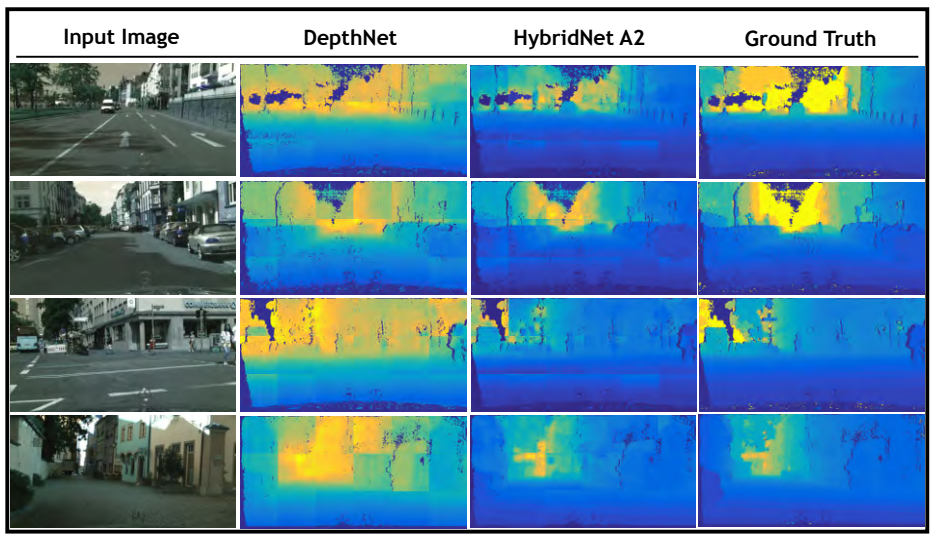



Figure 3.6: Depth estimation qualitative results. A visual comparison between the estimated depth maps against the ground truth is presented. In the first column is presented the input image, columns 2 and 3 depict the estimated depth maps obtained by DepthNet in [Iva16] and our hybrid model A2 respectively. Finally, ground truth is presented in column 4.

definition for these measures employed in the evaluation process. $d$ and $d^*$ represent the estimated depth and ground truth respectively. $N$ stands for the number of pixels with valid depth value in the ground truth depth map.

In the quantitative experiment, we compare the proposed hybrid architectures and DepthNet. Table 3.3 shows the quantitative results of the proposed hybrid architectures and DepthNet under the different evaluation measures introduced above. 9 metrics are used

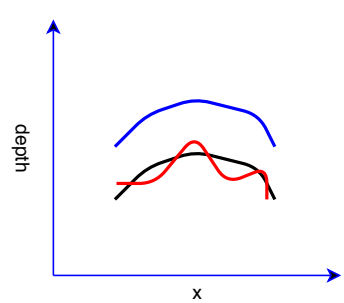



Figure 3.7: A 2D example of failures in depth estimation metrics

| | HybridNet A2 | HybridNet A1 | DepthNet [Iva16] | |
|---|---|---|---|---|
| $\gamma < 1.25$ (MVN) | **0.7483** | 0.6834 | 0.7248 | |
| $\gamma < 1.25$ | 0.5968 | 0.5037 | **0.6048** | higher |
| $\gamma < 1.25^2$ | **0.8221** | 0.8172 | 0.8187 | is |
| $\gamma < 1.25^3$ | **0.9292** | 0.9194 | 0.9152 | better |
| ARD | 0.24 | 0.2879 | **0.23** | |
| SRD | **4.27** | 4.35 | 4.43 | lower |
| RMSE-linear | **12.09** | 12.67 | 12.35 | is |
| RMSE-log | 0.4343 | **0.3407** | 0.4340 | better |
| SIE | **0.19** | 0.2 | 0.25 | |

Table 3.3: Quantitative evaluation of depth estimation results using metrics PP, PP-MVN, ARD, SRD, RMSE-linear, RMSE-log and SIE.

in the evaluation of depth estimation performance, because none of them is representative enough to absolutely ensure a fair comparison between different methods. For instance, Fig.3.7 shows a 2D example of two estimated depth values of a local patch in blue and red, and the ground truth in black. The blue estimation has a more similar shape to the ground truth but globally shifted, while the red estimation are less similar to the ground truth but laying closer to the ground truth. In the common sense, the blue estimation should be evaluated as the better estimation, however, in metrics like Absolute Relative Difference (ARD) and Square Relative Difference (SRD), the red estimation performs better. Therefore, we compare the performance between different approaches over all 9 metrics.

HybridNet A2 outperforms in 6 out of the 9 measures, which proves that training the feature extraction network for the tasks of semantic segmentation and depth estimation simultaneously improves also the depth estimation results. The comparison between Hybrid A2 and Hybrid A1 shows the necessity of clarifying the common and specific attributes of different tasks. Sharing only the common attributes of tasks in the feature extraction process leads to a better performance in depth estimation.

### 3.5.2 Indoor Scene

Road scene images have relatively limited variation in terms of the involved semantics and their spatial arrangements. It is usually captured by a camera fixed on a moving vehicle where the view direction of the camera is always parallel to the ground. This limits the variability of road scene images and makes it easier for the convolutional networks to learn to segment them robustly. In comparison, images of indoor scenes are more complex due to the free view point, the larger number of semantics in the scene, widely varying sizes of objects and their various spatial arrangements. On the other hand, although indoor scenes have smaller depth range than road scenes, it usually have more complex spatial layout, which provides challenges for depth estimation.

In this section, we evaluate the proposed architectures on indoor scene data for both semantic segmentation and depth estimation. We employ RGB-D Scene Understanding Benchmark Dataset (SUN-RGBD) [SLX15] for the experiments. SUN-RGBD contains over 10k RGB-D images of indoor scenes captured by 4 types of depth sensors, including also RGB-D images from NYU depth v2[SHKF12], Berkeley B3DO[JKJ+13], and SUN3D[XOT13]. It provides 2D ground truth object labels for 37 indoor scene classes, such as wall, floor, ceiling, table, chair etc and depth maps of different resolutions. Our task is to segment the objects within these 37 classes in each image while estimating the depth of it. In practice, we split the dataset into 5285 training and 5050 testing images, following the experiment configuration introduced in [BKC17].

Similarly to the experiments in Cityscapes dataset, we perform training data augmentation by random cropping, flipping and mirroring the original training images. However, in the testing phase, instead of cropping the test image like we did in the Cityscapes dataset, we downsample the test image to fit the input size of the hybrid architecture. Since the difference between the size of the test image and input size is not large in SUN-RGBD dataset, directly downsampling the test image to fit the input size strongly improves the efficiency

in the testing phase, while not leading the lost of important information in the test data.

### Semantic Segmentation

SUN-RGBD is a very challenging indoor scene dataset for semantic segmentation, in which object classes come in various shapes, sizes, different poses. There are also frequent partial occlusions between objects, which is typical in indoor scenes, due to the fact that many object classes are presented in each of the test images. In the proposed architectures, we use only the RGB modality as the input data for both training and testing. Fig.3.8 provides visual comparison for the estimated segmentation mask against ground truth. The figure presents, row-wise, 7 out-of-training examples where the first row shows the input images, the 2nd and 3rd row show the estimated segmentation mask from HybridNet A2 and DeepLab-ASPP respectively, and the last row shows the ground truth. HybridNet A2 shows stronger performance in distinguishing different objects in indoor scenes compared to DeepLab-ASPP.

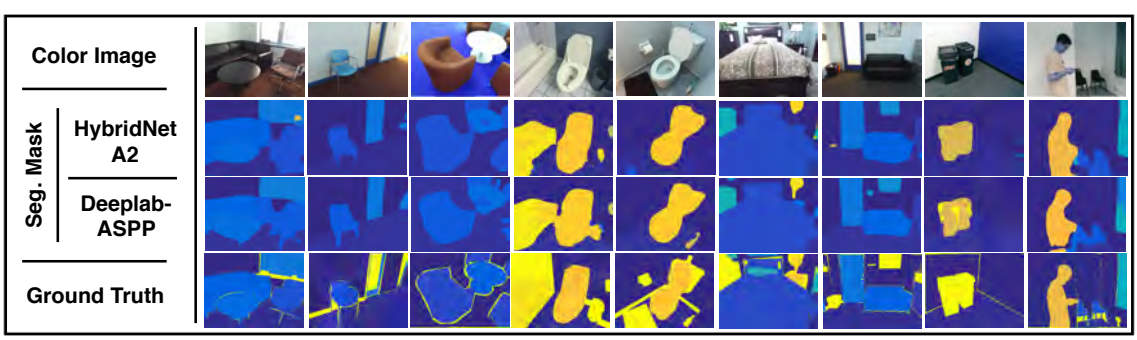



Figure 3.8: Semantic segmentation qualitative results. A comparison between semantic segmentation estimations against ground truth is presented. Input image is depicted in the first row. In the 2nd and 3rd are presented the estimated segmentation mask obtained from HybirNet A2 and the ground truth respectively.

Additionally to qualitative results, we follow the three measures introduced in Section 3.5.1: the Global Accuracy (G), the Class Average Accuracy (C) and mean Intersection over Union (mIoU) to evaluate the segmentation performance quantitatively. We also bench-

mark the proposed architectures against several other well adopted architectures for semantic segmentation, such as Fully Convolutional Network (FCN) [LSD15], SegNet[BKC17], DeepLab[CPK+16] and DeconvNet[NHH15]. For FCN, the parameters for the deconvolutional layers are learned from the training process instead of using fixed parameters to perform bilinear upsampling. For DeepLab, three architectures are employed, which are DeepLab-ASPP, DeepLab-LargeFOV and DeepLab-LargeFOV-denseCRF. They use the same VGGNet architecture for feature map extraction, which is similar to the proposed architectures. DeepLab-LargeFOV performs single scale upsampling on the feature map, while DeepLab-ASPP performs multi-scales upsampling, called Atrous Spatial Pyramid Pooling (ASPP). DeepLab-LargeFOV-denseCRF introduces a dense Conditional Random Field as a post-processing step for DeepLab-LargeFOV. Table 3.4 shows the quantitative results of the proposed architectures (HybridNet A1 and A2) compared with other methods. HybridNet A2 achieves the best results in C and mIoU over all the 7 methods while also obtain similar result (71.63%) in G than the best (73.87%) obtained in DeepLab-ASPP. The improvements against DeepLab-ASPP verifies again the idea of the multi-tasks learning, that estimating depth in addition to semantic segmentation helps the segmentation task (6.1% and 5.1% improvements in C and mIoU respectively). The performance of HybridNet A1 is even worse than the single task method DeepLab-ASPP, which indicates that the benefit of unifying two single tasks in a hybrid architecture can hardly be achieved by simply sharing the feature extraction process in more complex indoor scenes. The best segmentation performance obtained by HybridNet A2 compared with HybridNet A1 shows the importance of selecting a suitable unifying strategy in a multi-task learning problem and verifies the efficiency of the strategy employed in HybridNet A2.

| | G | C | mIoU |
|---|---|---|---|
| HybridNet A2 | 71.63 | **46.20** | **34.30** |
| HybridNet A1 | 69.34 | 38.64 | 28.68 |
| DeepLab-ASPP[CPK+16] | **73.87** | 40.09 | 29.22 |
| SegNet[BKC17] | 72.63 | 44.76 | 31.84 |
| DeepLab-LargeFOV[CPK+16] | 71.90 | 42.21 | 32.08 |
| DeepLab-LargeFOV-denseCRF[CPK+16] | 66.96 | 33.06 | 24.13 |
| FCN(learnt deconv)[LSD15] | 68.18 | 38.41 | 27.39 |
| DeconvNet[NHH15] | 66.13 | 32.28 | 22.57 |

Table 3.4: Quantitative evaluation of semantic segmentation results

### Depth Estimation

For depth estimation evaluation we present Fig.3.9 where a qualitative analysis of the results is depicted. The figure presents, column-wisely, the same 7 out of training examples presented in Fig.3.8, where the first row shows the input images, the 2nd and 3rd row show the estimated depth map from HybridNet A2 and DeepLab-ASPP respectively, and the last row shows the ground truth. The depth maps estimated by HybridNet A2 are more consistent with the ground truth than those obtained by DepthNet in terms of the depth layering.

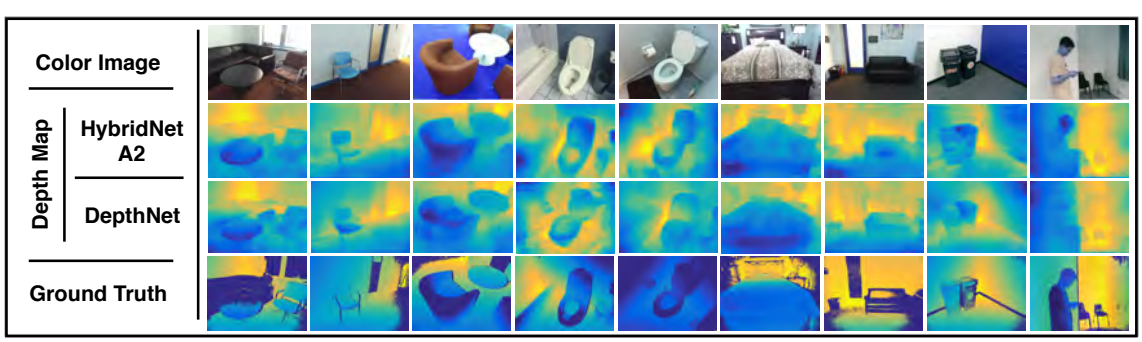



Figure 3.9: Depth estimation qualitative results. A comparison between depth estimations against ground truth is presented. Input image is depicted in the first row. In the 2nd and 3rd are presented the estimated depth map of our method and the ground truth respectively.

| | HybridNet A2 | HybridNet A1 | DepthNet | |
|---|---|---|---|---|
| $\gamma < 1.25$ (MVN) | **89.63** | 62.81 | 83.59 | |
| $\gamma < 1.25$ | **61.33** | 38.63 | 57.73 | higher |
| $\gamma < 1.25^2$ | **89.17** | 69.38 | 87.42 | is |
| $\gamma < 1.25^3$ | **97.43** | 86.28 | 97.08 | better |
| ARD | **0.202** | 0.301 | 0.218 | |
| SRD | **0.186** | 3.02 | 0.204 | lower |
| RMSE-linear | **0.682** | 8.35 | 0.715 | is |
| RMSE-log | **0.25** | 0.432 | 0.27 | better |
| SIE | **0.122** | 0.316 | 0.126 | |

Table 3.5: Quantitative evaluation of depth estimation results using metrics PP, PP-MVN, ARD, SRD, RMSE-linear, RMSE-log and SIE.

Additionally to qualitative analysis, we evaluate the performance following the measures introduced in Section 3.5.1: Percentage of Pixel (PP), Mean Variance Normalized Pixel of Percentage (PP-MVN), Absolute Relative Difference (ARD), Square Relative Difference (SRD), Linear Root Mean Square Error (RMSE-linear), Log Root Mean Square Error (RMSE-log) and Scale Invariant Error (SIE). Table 3.5 shows the quantitative results of the proposed architectures (HybridNet A1 and A2) and DepthNet under different measures. HybridNet A2 outperforms over all the measures, which proves that performing semantic segmentation in addition to depth estimation helps depth estimation task. The better performance of HybridNet A2 than A1 confirms the efficiency of the unifying strategy proposed in HybridNet A2 in more complex indoor scenes.

### Comparison with Other Hybrid Architectures

To compare HybridNet A2 with other hybrid architectures in the state-of-the-art, method proposed in [EF15] is employed. The method proposed in [EF15] addresses three different tasks including semantic segmentation, depth estimation and surface normal estimation. The architecture is designed as a stacking of three VGG structure [SZ14] representing different scales of feature extraction (shown in Figure 3.10). Each of the VGG structures takes

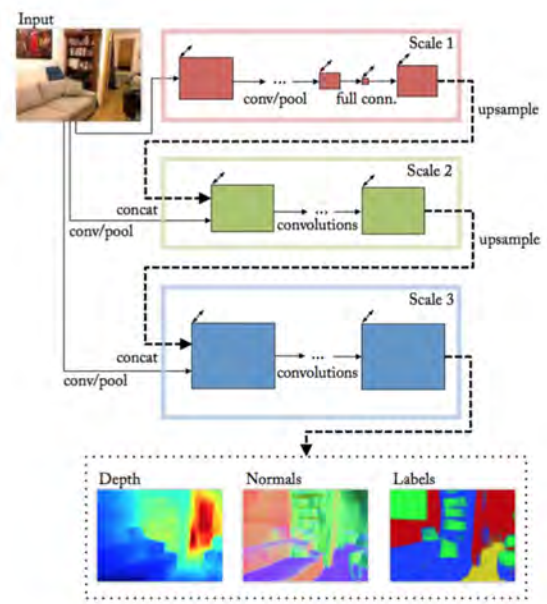



Figure 3.10: The hybrid architecture proposed in Eigen [EF15]

the output of the previous one along with the input color image as its input. Among the three tasks, depth estimation and surface normal estimation are two tasks tackled jointly, which means that these two tasks share the network in scale 1 while the networks in scale 2-3 are separately assembled for each task. For the semantic segmentation task, the architecture shown in Figure 3.10 is used again. But different from the other two tasks, the architecture of semantic segmentation allows two additional input channels which are depth and normal channels. This architecture is only fine-tuned from the previously trained hybrid model to generate semantic segmentation mask.

Although the source code of this method was not available, the performance evaluation is reported in a public dataset (NYU Depth V2 dataset [SHKF12]). To make the comparison with this approach, we trained and evaluated the our approach on NYU Depth V2 dataset. This data set includes RGB images and their corresponding 2D ground truth object labels for 40 indoor scene classes and depth map. NYU depth V2 dataset is divided into 795 images for training and 654 for testing. Due to the small amount of images available for

| | G | C | mIoU |
|---|---|---|---|
| HybridNet A2 | 64.7 | **48.4** | **36.5** |
| Eigen [EF15] | **65.6** | 45.1 | 34.1 |

Table 3.6: Quantitative segmentation results on NYU V2: G, C and mIoU

training, we augment the training set by random cropping, flipping and mirroring.

Tables 3.6 and 3.7 show the quantitative results of HybribNet A2 for both tasks and provides a comparison with the approach proposed in [EF15], denoted as Eigen. Semantic segmentation results in table 3.6 shows that HybridNet A2 outperforms Eigen in Class Average Accuracy (C) and mean Intersection over Union (mIoU) while keeping similar results than Eigen in Global Accuracy (G). It also illustrates that addressing RGB-D based semantic segmentation task under a multi-tasks learning schema better utilizes the depth information than directly feeding the depth information to the network as an extra input channel. On the another hand, depth estimation results in table 3.7 show that HybridNet A2 has a better performances in the relative measure SIE, while in the absolute measures Eigen outperforms HybridNet A2. The better performance of HybridNet A2 in the relative measure shows that HybridNet A2 has the better depth layering capability than Eigen, which is more concerned in the real applications. For absolute measures, we believe that the worse performance of HybridNet A2 is due to the weaker ability in describing the global layout of the scene. HybridNet A2 employs a much simpler architecture (AlexNet structure) for global depth network compared with the network of Scale 1 (VGG structure) in Eigen.

## 3.6 Conclusions

In this chapter, we have introduced a methodology for depth estimation and semantic segmentation from a single image using a unified convolutional network. The main goal of the proposed method is to seek for a better hybrid architecture of convolutional neural

| | HybridNet A2 | Eigen [EF15] |
|---|---|---|
| $\gamma < 1.25$ (MVN) | 0.7293 | - |
| SIE | **0.1571** | 0.171 |
| $\gamma < 1.25$ | 0.5006 | **0.769** |
| $\gamma < 1.25^2$ | 0.8013 | **0.95** |
| $\gamma < 1.25^3$ | 0.9422 | **0.98** |
| ARD | 0.2787 | **0.158** |
| SRD | 0.3236 | **0.121** |
| RMSE-linear | 0.9423 | **0.64** |
| RMSE-log | 0.3219 | **0.214** |

Table 3.7: Quantitative evaluation of depth estimation results on NYU V2 using metrics PP, PP-MVN, ARD, SRD, RMSE-linear, RMSE-log and SIE.

networks that modularizes the features extraction process by separating it into distinct features extraction for a specific task and common features extraction for both tasks. In this manner, both tasks can benefit from the extracted common features without being affected by those features only relevant to one task, which leads to a better performance. We also prove that solving correlated tasks like semantic segmentation and depth estimation together can improve the performance of methods tackling the tasks separately.

The qualitative and quantitative results shown in Section 3.5 demonstrate that unifying strategy employed in HybridNet A2 producing a better hybrid architecture for semantic segmentation and depth estimation compared to Hybrid A1. Hybrid A2 outperforms the results obtained by single task approaches, which proves that sharing underlying features extraction helps to improve the final performance in both tasks. Likewise, it is also proved that our methodology obtains comparable results than those benchmarking hybrid approaches.

Chapter 4

# Generic Instance Segmentation using RGB-D Stream Data

## 4.1 Introduction

While outstanding results are achieved in semantic segmentation by state-of-the-art supervised approaches, especially by Convolutional Neural Networks (CNNs) based approaches, there are still some drawbacks which may limit its application in higher level applications. First of all, the concept of semantic segmentation restricts the approaches to several predefined semantics, which means only the objects with these semantics will be segmented properly. Secondly, semantic segmentation usually requires a large amount of annotated data for training a classifier, in order to obtain semantic labels at pixel level. This compromises the application of these methods to RGB-D data, in which annotations are usually not sufficiently provided. Apart from that, semantic segmentation is also not aware of object instances, which makes it hard to apply to instance level applications.

From the other perspective, most of the unsupervised image segmentation approaches naturally have the advantage on coping with generic scenes, since no specific semantics are introduced in those segmentation systems. However, it is difficult to obtain object level segmentation based on all low level features extracted from a single image. More

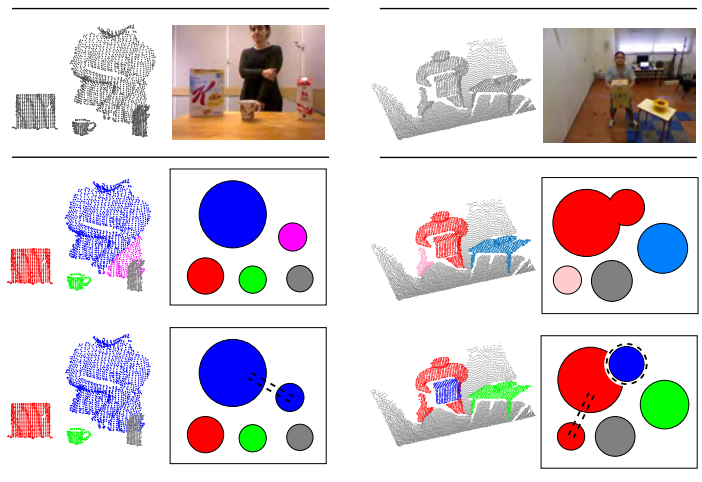
information is needed to bridge the gap between low level features and high level semantics in unsupervised approaches.

In this chapter, we explore the possibility of obtaining object level instance segmentation in generic scenes by introducing depth and temporal information from RGB-D stream data. We present a novel generic segmentation approach for 3D point cloud video (stream data) thoroughly exploiting the explicit geometry in RGB-D. Our proposal is only based on low level features, such as connectivity and compactness, which keeps the genericity of the approach. In practice, we employ a graph representation for 3D point clouds obtained from RGB-D stream data. The segmentation problem is then tackled based on energy minimization in the graph following the graph based image segmentation schema. On the other hand, we exploit temporal coherence by representing the rough segmentation of object instances in a single frame with a hierarchical structure, and propagating this hierarchy along time. The hierarchical structure provides an efficient way to establish temporal correspondences at different scales of object-connectivity, and to temporally manage the splits and merges of object instances. This allows updating the segmentation according to the evidence observed in the history.

## 4.2 Our Proposal

In Chapter 2, we have reviewed the related unsupervised approaches and study the limitations of them. In this section, we briefly explain our proposal, which works for generic instance segmentation in RGB-D videos.

We propose a generic instance segmentation method that works with 3D point clouds obtained from RGB-D stream data. It fully exploits the 3D geometry and temporal information in order to extract video objects and analyze their interaction in an unsupervised way. The proposed segmentation approach is *generic*, as it defines *objects* as "*compact point clouds*" in the 3D-space plus time domain. It allows point clouds corresponding to an object



(a) (b)

Figure 4.1: First row: input point clouds and color images. Second row: segmentation errors (false split in purple on left, false merge in red on the right) in challenging scenes with occlusions/self-occlusions or object interactions, and the corresponding sketch maps shown besides. Third row: Our segmentation result by analyzing generic features in spatio-temporal domain to handle the challenges without introducing neither strong prior knowledge nor initialization, together with the corresponding sketch maps.

to break into different compact sub-clouds due to occlusions, or to merge with point clouds corresponding to other objects when they become spatially close (this is what we call *object interaction*). Fig. 4.1(a) shows an example where the *object* "human" breaks into two compact clouds (blue and purple) due to self-occlusion, while Fig. 4.1(b) shows an example where the objects "human" and "box" become spatially close and merge in a compact point cloud (in red).

We propose a *hierarchical representation* of the raw point cloud data to cope with these situations considering 3D spatial connectivity, and exploit the temporal information by building the *temporal correspondences* between the hierarchical structures in successive frames. But, in contrast to [HBEC14, GKHE10], we do not construct the hierarchy from a relatively fine level. We rather start at a much higher level formed by blobs, segments, com-

ponents and objects, as explained later in Section 4.5, so that the task of building temporal correspondences can be solved globally as an optimization problem thanks to the reduced problem scale. Building temporal correspondences at a higher level does not affect object segmentation, since only object correspondences are concerned rather than object details in the segmentation task. Then, based on the established temporal correspondences, objects in a given frame are defined according to the evidence observed up to this frame.

The rest of the chapter is organized as follows: in section 4.4 we define point cloud connectivity and the detection of compact point clouds in a single frame. Section 4.5 presents the framework for the temporally coherent segmentation approach based on compact point clouds. Results are shown in section 4.6, and section 4.7 discusses the results and yields conclusions.

## 4.3 Point Cloud Acquisition

A consumer depth sensor, like Kinect or Asus Xtion etc, usually captures a depth map registered with a color image at a time, called an **RGB-D Frame**. Each pixel in an RGB-D frame can be represented by a vector $\langle R, G, B, x, y, d \rangle$, where $R, G, B$ stands for the photometric information provided by the color image, $x, y$ stands for the coordinate of the pixel on the frame and $d$ stands for the distance from the image plane to the 3D point in the scene represented by this pixel. With intrinsic parameters of the camera, $\langle x, y, d \rangle$ can be transformed to a 3D point $\langle X, Y, Z \rangle$ in the real world coordinate system. In this thesis, a **Point Cloud** is defined as a set of 3D points with real world coordinates. Transforming all the pixels produces the point cloud of the scene. It corresponds to a sampling of the visible scene surface from the camera viewpoint and represents the 3D geometry of the scene surface discretely by the groups of 3D points.

## 4.4 Single Frame Compact Point Cloud Detection

An intuitive idea to achieve segmentation on a point cloud of a scene is to detect "compact point clouds" sub-clouds, in which compact point clouds are considered to be objects with respect to the geometry of the scene, since object instances in the scene are usually solid. To evaluate if a point cloud is compact or not, it is necessary to define the 3D spatial connectivity in a point cloud.

### 4.4.1 Spatial Connectivity in Point Clouds

Unlike the well organized space of image coordinates, a point cloud is a set of scattered 3D points, where the adjacency between 3D points can not be measured directly. Spatial connectivity among those 3D points can simply be defined by a distance threshold. Point pairs with the distance lower than the threshold are treated as neighbors. Another way to define point cloud connectivity is to voxelize the 3D space. In this case, a 3D grid is created for the space. Points on the point cloud are mapped into 3D grids (voxels) and points in the same voxel are represented by the centroid of the voxel in the voxelized point cloud. The adjacency is then defined as the adjacency on the 3D grid.

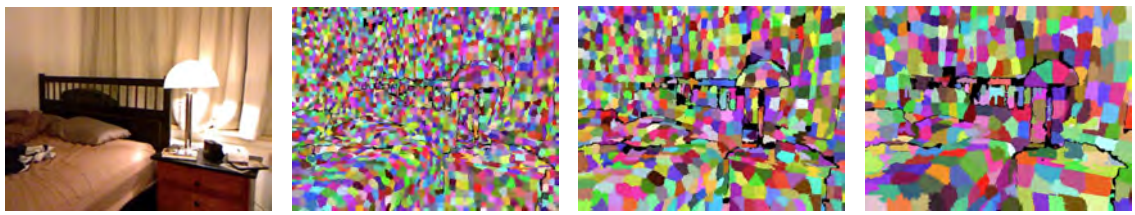
Point cloud connectivity defined on 3D points or voxels would usually provide an excess of connectivity information which is not necessary for object segmentation and would strongly enlarge the data structure. On the other hand, point clouds obtained from a single RGB-D camera have a significant drawback: point density, i.e. details available about scene geometry, falls rapidly with increasing distance from the camera. This also prevents these definitions of spatial connectivity.

In the proposed approach, we follow the method introduced in [CSSPW14] and [PKAW13] to robustly construct spatial connectivity for point clouds. We first compensate the decreasing point density and quantization with increasing depth by using the coordinate transformation in [CSSPW14]. Next, we build a super-voxel representation [PKAW13] on the

transformed point cloud. When building super-voxels of a point cloud, a grid voxel filtering step is first performed to organize the 3D space into voxel grids. In this manner, noise in a point cloud is somehow reduced by representing the points in a voxel with the voxel center. The spatial connectivity is defined at the super-voxel level, rather than on the raw point cloud. This also allows to start the segmentation from a higher level.

In practice, given a point cloud $C$, we transform $C$ with the reversible transformation $T(C) \rightarrow C' : x' = x/z, y' = y/z, z' = log\,(z)$. The division of the $x$ and $y$ coordinates by $z$ compensates for the perspective transformation [CSSPW14], equalizing the point density in the $x-y-$plane across the depth range. Transforming the $z$ coordinate helps to deal with the effects of depth quantization by compressing points as depth increases.

The transformed point cloud $C'$ is organized using a 3D grid, and 3D points are mapped to voxels in the voxelized cloud. In order to define the point cloud connectivity at a higher level, we group voxels into locally homogeneous patches in the 3D space, called super-voxels. Voxels are grouped in a 39 dimensional feature space $\langle L, A, B, X, Y, Z, FPFH_{1...33} \rangle$, considering 1) their distance in 3D space, 2) their color similarity, and 3) local 3D shape similarity. The color of each voxel is represented by the mean color in CIELab color space of the points in that voxel grid. The coordinate of each voxel is set to the centroid of it. To measure the shape similarity, the local 3D shape of each voxel is represented by Fast Point Feature Histograms (FPFH) introduced in [RHBB09]. In practice, a group of seeds is first uniformly selected in the 3D space with a density factor $R_{seed}$ as the initial clusters for creating super-voxels. Then, voxels are clustered into different initial clusters iteratively using a local k-means approach related to [ASS$^+$12, WGB12] in the 39 dimensional feature space. Fig.4.2 shows the super-voxels generated under different seed density. We finally represent the transformed point cloud $C'$ as a super-voxel graph $G\,(v, e)$, in which nodes $v_i \in v$ are super-voxels (homogeneous sub-cloud patches), and edges $e_{i,j}\,(v_i, v_j) \in e$ define the adjacency of patches. In this manner, a point cloud is simplified as a graph of super-



(a) (b) (c) (d)

Figure 4.2: An example of the super-voxels generated in our approach from a point cloud with different seed density. (a) The color image, (b)$R_{seed} = 0.06m$, (c)$R_{seed} = 0.1m$, (d)$R_{seed} = 0.15m$. Each super-voxel is labeled with a random color.

(a) (b)

Figure 4.3: An example of the super-voxels generated in our approach from a point cloud. (a) The original point cloud. (b) The super-voxels (each super-voxel is labeled with a random color).

voxels where important boundary information is kept.

The spatial connectivity in a point cloud is therefore dealt with as the connectivity among the super-voxels in the graph. Fig. 4.3 shows an example of the super-voxels generated from a point cloud. The spatial connectivity built on super-voxels represents the geometry of the scene finely enough. We also evaluated different graph building methods in Section 4.6, concluding that the method in [PASW13] outperforms other methods such as [SRHC13] for an object segmentation task.

Based on the 3D spatial connectivity, we define **Compact Point Cloud** as a point cloud represented by a set of super-voxels, when the graph built with this set of super-voxels is a connected graph.

### 4.4.2 Compact Point Cloud Detection

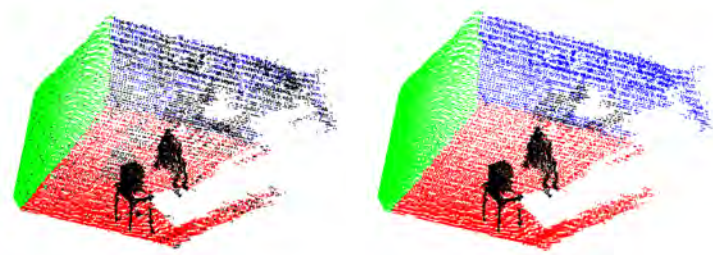
Since a point cloud and its spatial connectivity are represented by a super-voxel graph, the most intuitive way to detect compact sub-clouds of it is to search for the connected components of the graph. However, the detected compact sub-clouds in a single frame will not usually correspond to objects, due to occlusions, object interactions and connections to the supporting plane, as shown in Fig. 4.1. Unless prior knowledge is introduced, it is difficult to segment objects by detecting compact sub-clouds of a point cloud in a static image. Instead, we attempt to obtain, for each single frame, those connected components that are not part of the supporting planes in the whole graph of the scene. The obtained connected components are our first approximation of the objects in the scene, which will then be refined by exploiting temporal information: splits, merges and object interactions.

In practice, we build a super-voxel graph to represent the point cloud in each frame as explained in Section 4.4.1. Then, we apply a plane detection technique to eliminate large plane shaped regions that may correspond to supporting plane in the point cloud. Finally, we extract connected components in the remainder of the graph by analyzing spatial connectivity. The refinement for this first approximation of objects using temporal information is introduced in Section 4.5.

#### Detection of Supporting Planes

In most scenes, foreground objects are captured together with (spatially connected) background such as floor, ceiling or walls. The background point cloud serving as supporting planes for foreground objects usually connects isolated foreground objects, thus preventing segmentation via connectivity analysis. Therefore, we remove large plane shaped regions before analyzing the connectivity in the point cloud by applying a 3D plane detection technique on the point cloud.

A plane detection method was proposed in [HHRB11], which categorizes the points on



(a) (b)

Figure 4.4: An example of plane detection results. The points in red, green and blue belong to detected planes. The points in black are the points in the "not on any plane" class. (a) Plane detection result from [HHRB11]. (b) Plane detection result from our approach.

the cloud into $n+1$ classes corresponding to **plane 1...n** and "**not on any plane**" in two steps. First, a local surface normal vector for each point on the cloud is estimated by finding two vectors which are tangential to the local surface within the neighborhood of this point on the image plane. From these two tangential vectors, the normal is computed using the cross product. Then, these points are clustered in a voxelized normal space in order to obtain clusters of points with similar local surface normal orientation, while discarding clusters of small size. Secondly, each of the obtained clusters is split into plane clusters, so that each of the new clusters resembles a single plane. Thus, for each point in a cluster, the distance between the origin to the plane crossing this point with the averaged and normalized normal of this cluster is computed. Then a similar clustering step is applied in the distance space to classify points into different plane clusters. The obtained $n$ plane clusters represent the detected $n$ classes of planes, and the discarded clusters form the **not on any plane** class.

However, simply applying the plane detection method in [HHRB11] may fail when the point cloud is noisy (see Fig. 4.4(a)). So, based on the plane detection result, we build another layer modeled as a Conditional Random Field (CRF) [1] on the super-voxel graph to

[1]Traditional CRFs models involve a unary energy $\mu$ and a pairwise energy $\rho$, which respectively represent the degree that one node belongs to a label and the strength of an edge connecting two nodes.

provide extra robustness to the noise in the point cloud considering its spatial connectivity. In this layer, we propose to label the nodes (super-voxels) in the graph with the $n+1$ labels using a unary data energy, defined on the basis of the detected planes, and a pairwise smoothness energy, defined on the basis of the graph structure. The energy in the CRFs model is then optimized via graph cut [BK04] using alpha expansion [BVZ01] to obtain the best labelling for the graph.

In practice, given the $n+1$ classes for points on a cloud obtained from the plane detection result and its super-voxel graph $G$, the energy function, $E^p(\cdot)$, is formulated as the summation of the unary data energy $\mu^p$ and the pairwise smoothness energy $\rho^p$. In Eqs.4.1,4.2 and 4.3, $v_i \in v$ and $e_{i,j}(v_i, v_j) \in e$ stand for a node and an edge on the graph $G(v, e)$ respectively, $l^p$ stands for a labeling that assigns each node $v_i \in v$ a label $l_{v_i}$ in the label set $L^p = \{1, ..., n+1\}$:

$$E^p(l^p) = \underbrace{\sum_{v_i \in v} \mu^p_{v_i}(l_{v_i})}_{unary\ energy} + \underbrace{\sum_{(v_i,v_j)\in e} \rho^p_{v_i,v_j}\left(l_{v_i}, l_{v_j}\right)}_{pairwise\ energy} \tag{4.1}$$

The unary data energy measures the cost of assigning $l_{v_i}$ to node $v_i$ given the observed data. In our case, it depends on the percentage of points in point cloud $C_{v_i}$ which are clustered to class $l_{v_i}$ in the plane detection process, denoted as $C_{l_{v_i}}$. $NoP(C)$ computes the number of points in a point cloud $C$.

$$\mu^p_{v_i}(l_{v_i}) = 1 - \frac{NoP\left(C_{l_{v_i}}\right)}{NoP(C_{v_i})} \tag{4.2}$$

The pairwise smoothness energy specifies the cost of assigning different labels to $v_i$ and $v_j$ connected by $e_{i,j}$. It is defined as the cosine similarity between the normals of these two nodes ($\overrightarrow{N_{v_i}}$ and $\overrightarrow{N_{v_j}}$), since edges connecting nodes with high normal difference usually coincide with boundaries. Note that the normal of a node (super-voxel) is computed by

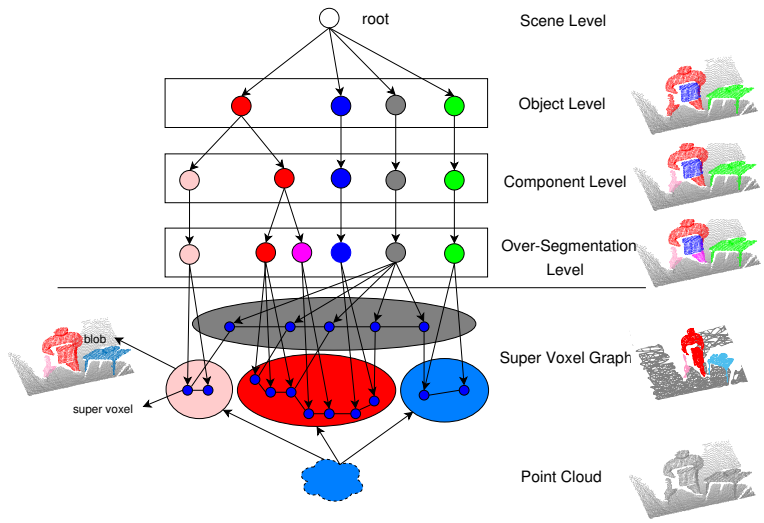



Figure 4.5: The hierarchical structure built for a point cloud. Different nodes at the same level in the hierarchy are labeled with different colors. The point cloud beside it is labeled with the same color of its related node.

averaging the normals estimated with [HHRB11] at points belonging to this super-voxel.

$$\rho^{p}_{v_i,v_j}\left(l_{v_i}, l_{v_j}\right) = \begin{cases} max\left(\frac{\overrightarrow{N_{v_i}}\cdot\overrightarrow{N_{v_j}}}{\left\|\overrightarrow{N_{v_i}}\right\|\cdot\left\|\overrightarrow{N_{v_j}}\right\|}, \frac{\overrightarrow{N_{v_i}}\cdot-\overrightarrow{N_{v_j}}}{\left\|\overrightarrow{N_{v_i}}\right\|\cdot\left\|\overrightarrow{N_{v_j}}\right\|}\right) & l_{v_i} \neq l_{v_j} \\ 0 & otherwise \end{cases} \tag{4.3}$$

The energy function in Eq. 4.1 is optimized via a graph cut method [BK04] to generate the best labelling. The nodes (super-voxels) labeled as $1...n$ form $n$ plane classes respectively. Fig. 4.4 compares an example of planes detected by the method in [HHRB11] and in our approach, showing higher robustness to noise in the point cloud.

### Extracting Connected Components

After removing the nodes labeled as $1..n$ in the graph, we build a new graph with the **"not on any plane"** nodes. We extract $m$ connected components on the new graph by analyzing the spatial connectivity. The lower part of Fig. 4.5 presents a simplified example of our single frame compact point cloud detection process, in which the blue circles and the edges between them stand for the graph representation built on the input point cloud. The ellipses marked in different colors show the detected compact point clouds in one frame. We denote those $n + m$ compact point clouds with the name "**blob**" in the rest of this paper. Note how the $m$ non-plane like blobs (in red, pink and blue) are spatially connected to the floor blob (in grey). The elimination of the floor allows us to obtain non-plane like blobs as connected components in the graph.

## 4.5 Temporally Coherent 3D Segmentation

In this section, we explain how the 3D point cloud video segmentation task is tackled by modelling the detected blobs in each frame with the proposed hierarchical structure and propagating this structure along time. The single frame compact point cloud detection method described in Section 4.4 exploits only the spatial connectivity in one frame to extract blobs, which ideally correspond to objects. In real cases, the point cloud corresponding to an object can split in different blobs due to occlusions/self-occlusions, or can merge in a single blob with the point cloud corresponding to other objects due to what we call object interactions. In a single frame analysis, it is difficult to produce proper object segmentation without introducing prior knowledge. To tackle the problem of object splits/merges while keeping our approach generic, we propose to introduce temporal coherence when a stream of RGB-D data is available. More specifically, we segment the detected blobs in each frame into meaningful sub-clouds and represent these sub-clouds in a hierarchical fashion. Then, we

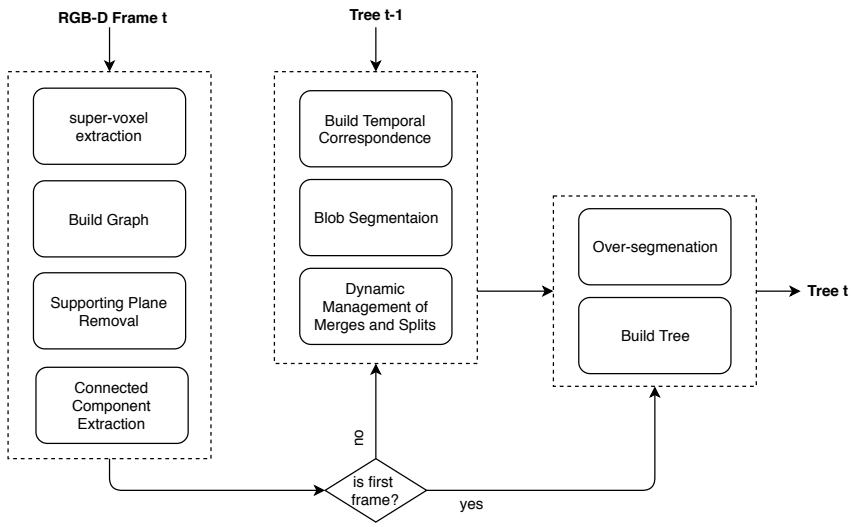



Figure 4.6: An overview of the proposed approach.

associate temporally these sub-clouds in the hierarchies in order to maintain the trajectories for them, and to analyze the correlation along time, always without explicit object models or accurate initialization to keep the genericity of our approach, and in order to make the best possible decision with the accumulated information at a given time. Fig.4.6 shows an overview of the proposed approach.

### 4.5.1 Hierarchical Representation

Before introducing the proposed hierarchical structure, let us first define the terms and concepts that we use. Given a super-voxel graph $G\,(v, e)$ at time $t$ and the object segmentation on it $\bigcup_{i=1}^{M_o} G_{o_i} = G$, $G_{o_k} \cap G_{o_q} = \emptyset$ for $k \neq q \in [1, M_o]$, **blobs** are the connected components on the super-voxel graph $G$, where $\bigcup_{i=1}^{M_b} G_{b_i} = G$, $G_{b_k} \cap G_{b_q} = \emptyset$ for $k \neq q \in [1, M_b]$. For each object $o_i$, $i \in [1, M_o]$, we define its **components** $c_j^i$, $j \in [1, M_c]$ as the connected components on $G_{o_i}$, where $\bigcup_{j=1}^{M_c} G_{c_j^i} = G_{o_i}$, $G_{c_k^i} \cap G_{c_q^i} = \emptyset$ for $k \neq q \in [1, M_c]$. For each component $c_j^i$, we over-segment it into **segments**, where $\bigcup_{u=1}^{M_s} G_{s_u^{i,j}} = G_{c_j^i}$, $G_{s_k^{i,j}} \cap G_{s_q^{i,j}} = \emptyset$ for $k \neq q \in [1, M_s]$.

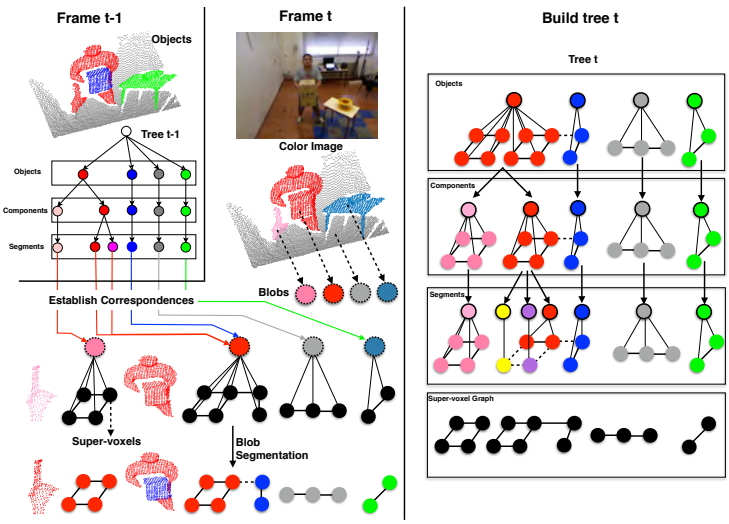



Figure 4.7: An example of hierarchical structure creation. Upper-left: Object segmentation at $t-1$ and its hierarchical structure; Middle column: color image at $t$, detected blobs in the point cloud at $t$, illustrations about the establishment of correspondences and blob segmentation process; Right: hierarchical structure building process at $t$.

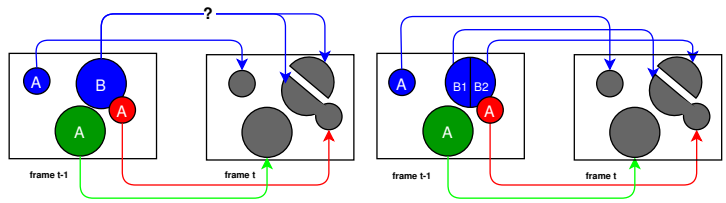
We build the hierarchical structure as a tree, in which 4 levels varying from coarse to fine represent the object segmentation at different scales of object-connectivity. The upper part of Fig. 4.5 shows the hierarchical structure for a point cloud. Note that colors are used to differentiate nodes at the same level in the hierarchy and the point clouds plotted beside it are marked in the same color of their related nodes. The root of the tree represents the **scene**. The second level of the tree is the **object level**, in which each node stands for one object in the object segmentation. The merges between objects are handled at this level by maintaining the similarities among objects along time. The next level, named **component level**, is employed to handle potential splits of point clouds representing these objects. Thus, an object is represented by more than one component when it splits in different blobs. Components from different objects can be part of the same blob, because of the interactions between objects. Splits of an object are managed by maintaining the

similarities among the components of this object along time. Managing object splits and merges in this manner provides a way to update the object segmentation according to the evidence observed up to time $t$. The final level of the tree is the **over-segmentation level**. We over-segment components into segments using normalized cut in their graphs in order to correctly establish correspondences between hierarchies along time and update its structure, that is, to obtain temporally coherent object labelling. However the amount of segments generated at this level (finest level in our hierarchy) is much less than the finest level employed in methods like [HBEC14, GKHE10].

### 4.5.2 Hierarchical Structure Creation

In Section 4.5.1, we have represented the detected blobs in one frame hierarchically, given the object segmentation of this frame. In this section, we explain how we obtain the object segmentation taking into account the objects segmented in the previous frame to create the hierarchical representation for the current frame. We model the segmentation task as a label assignment problem, in which we build temporal correspondences to label the super-voxels in the current frame considering the object labelling in the previous frame. But instead of following the method in [HBEC14, GKHE10, APP$^+$12] to build the temporal correspondences at a very fine level (super-voxels in our case), we propose to first relate the object labels in the previous hierarchy to the blobs detected in the current frame by minimizing an assignment energy defined on the difference of point cloud size and displacement. Then we split the blobs associated to more than one object label by modeling the problem as a multi-label segmentation task with a fully connected CRF [Kol11]. There are several advantages for this method:

- Object labels in the previous hierarchy are associated to blobs in the current frame via a small number of segments at the finest level, which strongly reduces the scale of the problem of building temporal correspondences




(a) (b)

Figure 4.8: An example of temporal inconsistency problem. (a) The problem when establishing the correspondences between components in the previous frame and blobs in the current frame. (b) Using the segments instead of components solves this problem.

- The task of building temporal correspondences can be solved globally as an optimization problem due to the reduced problem scale
- The temporal consistency problem can be easily addressed by generating sufficient segments at the over-segmentation level in the previous hierarchy

The object segmentation in the first frame is obtained by simply taking the detected blobs (i.e. connected components on the super-voxel graph) as the object segmentation, because no prior information whatsoever about the objects is provided. Accordingly, we create one component for each object and over-segment each component into segments.

### Establishing Temporal Correspondences

Apart from the first frame, we obtain the object segmentation in a temporal coherent way by considering both the blobs detected in the current frame and the object segmentation in the previous frame. Fig. 4.7 shows an example of how we create the hierarchy for frame $t$. The object segmentation at $t-1$ and its hierarchical representation are shown at the upper-left corner of Fig. 4.7. For the data in frame $t$, we show the color image and detected blobs (labeled in different colors) in the middle. Then the point clouds of the blobs are represented as circles with a dash border, each of which consists of several super-voxels (e.g. the red blob consists of 6 super-voxels, as shown below). We also show the corresponding

point cloud beside the blobs.

Our goal is to label the super-voxels in different blobs at frame $t$ with the object labels contained in the hierarchy at $t-1$, and to obtain the object segmentation at $t$ from this labelling. In our approach, a correspondence is made between the blobs in the current frame and the segments at the over-segmentation level of the hierarchy at $t-1$. This is a first step to warranty the temporal continuity of the segmentation in the video objects. Fig. 4.7 shows the label assignment process, in which object labels in hierarchy $t-1$ are associated to blob labels at $t$. The color of the lines, linking segments in hierarchy $t-1$ and blobs at $t$, correspond to the object nodes in hierarchy $t-1$, which indicates that object labels are associated to blobs via segments. The over-segmentation level in the tree is employed to tackle temporal consistency problems. Fig. 4.8(a) shows an example of this, where objects are marked in different colors and their components are denoted with letters. The component B of the blue object in frame $t-1$ splits into two blobs in frame $t$. In this case, no correct association is found between components at $t-1$ and blobs at $t$. The problem may be tackled by over-segmenting the component $B$ of the blue object into segments $B1$ and $B2$ (shown in Fig. 4.8(b)) and associating the segments in frame $t-1$ with blobs in frame $t$.

Given the $M_b$ blobs $b_i$ detected in frame $t$, $M_s$ segments $s_i$ in frame $t-1$ and their corresponding point clouds ($C_{b_i}$ and $C_{s_i}$), establishing the correspondence between the blobs and the segments is a problem of assigning $M_b$ blob labels to $M_s$ segments. This can be interpreted as a process of allocating balls (segments) into baskets (blobs), which finds the best allocation (temporal correspondences) between segments and blobs. In order to cope with possible segments moving out of the scene, we create a virtual empty blob $b_{out}$. This allows assigning $b_{out}$ to any segment in the correspondence building process, which implicitly represents the segments moving out of the scene. The labels assignment task is a nonlinear integer programming problem. We solve it using a Genetic Algorithm [Whi94] to minimize an energy function $E^{as}(\cdot)$, which is composed of three terms representing the appearance

changes $E^a$, the displacements $E^d$ and the penalty when objects move out of the scene $E^o$.

$$E^{as}\left(l^{as}\right) = E^a + E^d + E^o \tag{4.4}$$

where $l^{as}$ is an assignment proposal which assigns each segment $s_i$ a label $l_{s_i}$ in the label set $L^{as}$. $E^a$ stands for the overall appearance difference between each of the $M_b$ blobs $b_i$ and its corresponding segments $\left\{s_j \mid l_{s_j} = b_i\right\}$ under $l^{as}$. In practice, the appearance difference is defined as a size measure, by computing the difference on the number of points between them. $NoP\left(C\right)$ counts the number of points in the point cloud $C$.

$$E^a\left(l^{as}\right) = \sum_{i=1}^{M_b} \left| NoP\left(C_{b_i}\right) - \sum_{\left\{s_j | l_{s_j} = b_i\right\}} NoP\left(C_{s_j}\right) \right| \tag{4.5}$$

$E^d$ represents the overall displacement for moving each of the $M_s$ segments $s_j$ at $t-1$ to the location of its corresponding blob $b_i$ at $t$ under $l^{as}$. Specifically, we employ the Hausdorff distance $d_h\left(\cdot\right)$ to compute the displacement between point clouds.

$$E^d(l^{as}) = \sum_{i=1}^{M_b} \sum_{\left\{s_j | l_{s_j} = b_i\right\}} d_h\left(C_{s_j}, C_{b_i}\right) \tag{4.6}$$

$E^o$ stands for a penalty when segment $s_j$ moves out of the scene, that is, when $b_{out}$ is assigned to it. In this case, we calculate the Euclidean distance $d_e\left(\cdot\right)$ between the centroid of $C_{s_j}$ to the closest boundary among the predefined $M_{\mathbb{P}}$ boundaries $\mathbb{P}_i$, which are set with respect to the field of view of the camera.

$$E^o(l^{as}) = \sum_{\left\{s_j | l_{s_j} = b_{out}\right\}} \min_{i=1\ldots M_{\mathbb{P}}} \left(d_e\left(C_{s_j}, \mathbb{P}_i\right)\right) \tag{4.7}$$

### Blob Segmentation

In order to obtain the object segmentation, we still need to segment the blobs when segments corresponding to different objects in the previous frame are related to the same blob. For example, in Fig. 4.7, we perform segmentation in the red blob, where three segments that correspond to two different objects are related to it. Segmenting the red blob produces two partitions which respectively correspond to the red object and blue object in tree $t-1$. In this manner, each partition is related to only one object in the previous frame. These partitions and the blobs related with only one object label form the object segmentation for the current frame.

In our approach, we formulate the blob segmentation problem as a node labelling task on its related graph $G_b(v, e)$, in which nodes are super-voxels and edges show the adjacency of super-voxels. We label each of the super-voxels $v_i$ in the graph with object labels $o_i$ related to this blob. This is usually achieved by employing Conditional Random Field (CRF) models [RKB04, KT$^+$09]. Traditional CRF models involve a unary energy $\mu$ and a pairwise energy $\rho$, which respectively represent the degree that one node belongs to a label and the strength of an edge connecting two nodes. Minimizing the CRF energy produces the optimal labelling on the graph, that is also the segmentation of the blob. However, the pairwise energy is only computed for neighboring nodes on the graph in traditional CRF models, which makes the boundaries between different labels favor the "thinner" part of the graph with less edges. To overcome this limitation, we employ a fully connected CRF [Kol11] model in our system. In the fully connected CRF model, the pairwise energy is established on any pair of nodes on the graph, which makes the "shape" of the graph less critical to the optimal labelling of the graph. In practice, the energy function in the fully

connected CRF model is modeled as:

$$E^s\left(l^s\right) = \underbrace{\sum_{v_i \in v} \mu^s_{v_i}\left(l_{v_i}\right)}_{unary\ energy} + \underbrace{\sum_{(v_i,v_j)\in e} \rho^s_{v_i,v_j}\left(l_{v_i}, l_{v_j}\right)}_{pairwise\ energy} \tag{4.8}$$

where $l^s$ stands for the labelling which assigns each super-voxel $v_i$ a label $l_{v_i}$ in the label set $L^s$. We follow the unary energy defined in [LCP16], where the unary energy of labelling node $v_i$ with object label $o_j$, $\mu_{v_i^s}\left(l_i = o_j\right)$ is proportional to the mean distance between node $v_i$ in the current frame and the k-nearest nodes labeled by $o_j$ in the previous frame. For the pairwise energy, we extend the one defined in [Kol11] for nodes representing pixels on 2D images to an energy which is suitable for nodes representing 3D point clouds. Specifically, we adopt an appearance and an smoothness term balanced with weights $\omega_1$ and $\omega_2$. The appearance energy term is defined as the 3D Euclidean distance $d_e\left(C_{v_i}, C_{v_j}\right)$ between the centroids of two point clouds and the color distance $d_{rgb}\left(C_{v_i}, C_{v_j}\right)$ as the difference between the mean color of each point cloud in the Gaussian kernel $\exp\left(-\frac{d(\cdot)}{2\sigma^2}\right)$. The smoothness energy term is defined as the 3D Euclidean distance between the centroid of two point clouds in the Gaussian kernel.

$$\begin{aligned}
&\rho^s_{v_i,vj}\left(l_i, l_j\right) \\
&\quad = \begin{cases}
\omega_1 \exp\left(-\dfrac{d_e\left(C_{v_i}, C_{v_j}\right)}{2\sigma_\alpha^2} - \dfrac{d_{rgb}\left(C_{v_i}, C_{v_j}\right)}{2\sigma_\beta^2}\right) \\
\qquad + \omega_2 \exp\left(-\dfrac{d_e\left(C_{v_i}, C_{v_j}\right)}{2\sigma_\gamma^2}\right) & l_i \neq l_j \\
0 & otherwise
\end{cases}
\end{aligned} \tag{4.9}$$

As shown in Eq. 4.9, $\omega_1$ and $\omega_2$ are used to balance the appearance energy and smoothness energy. $\sigma_\alpha$, $\sigma_\beta$ and $\sigma_\gamma$ control the scale of the Gaussian kernel. The energy function in Eq. 4.8 is minimized using an efficient message passing implementation based on the mean

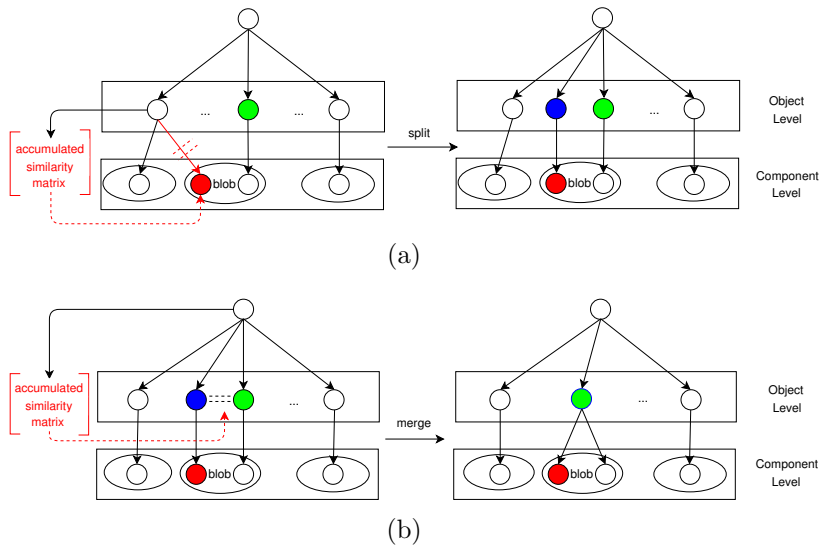



Figure 4.9: Example of how we update the object segmentation in the current frame by dynamically managing object splits (a) and merges (b)

fields approximation and high dimensional filtering [Kol11]. The optimum represents the best labelling on graph $G_b$, which also corresponds to the segmentation of the blob. Each partition in this segmentation is related to an object in the previous frame.

The object segmentation for the current frame is then formed by the partitions and blobs related with only one object label. Then, the hierarchy in the current frame is built based on the current object segmentation, starting from the object level to the component level following the criteria explained in Section 4.5.1 (see the right part in Fig. 4.7), while the correspondences between the current hierarchy and previous hierarchy are established in the object and component level accordingly.

### Dynamic Management of Merges and Splits

In the current hierarchy obtained in Section 4.5.2, we have the segmented objects for the current input data at the object level of the tree, which are temporally coherent with the segmented objects in the previous frame. However, they may not represent the proper object segmentation, since no accurate initialization is guaranteed at the beginning of this process

in our approach. That is to say, the segmented objects need to be further analyzed along the time, in order to cope with the errors in the previous information. In this case, we exploit the established correspondences and analyze the behaviors of related nodes in hierarchies along time. More precisely, we maintain similarities between nodes at the component and object level respectively and update the object segmentation in the current hierarchy based on it. The component similarities are measured among components belonging to the same object, while the object similarities are measured among all objects. These similarities are computed by considering the distances between the point clouds of components $C_c$ or objects $C_o$, which reveal the likelihood of object splits and merges. In our approach, the similarity between point cloud $C$ and $C^*$ is inversely proportional to the shortest distance between the two point clouds. The shortest distance $d_s(C, C^*)$ is measured based on the corresponding graphs ($G$ and $G^*$) built on $C$ and $C^*$, in which we search for the shortest distance between nodes $v_i$ in $G$ and nodes $v_j$ in $G^*$. $\psi$ is a normalizing factor which normalizes distances between two point cloud smaller than $\psi$ while forcing the similarity between point clouds equal to zero when the distance is larger than $\psi$.

$$Sim(C, C^*) = \begin{cases} 0 & d_s(C, C^*) > \psi \\ 1 - \frac{d_s(C, C^*)}{\psi} & otherwise \end{cases} \tag{4.10}$$

$$d_s(C, C^*) = \min_{v_i \in G, v_j \in G^*} d_e(v_i, v_j) \tag{4.11}$$

We accumulate the similarities along time by averaging the current similarity and the previous accumulated similarity using the correspondences built at component and object level between trees. The accumulated similarity reveals the likelihood of object splits and merges regarding the evidences observed up to the current frame. Then object splits and merges are confirmed by thresholding the accumulated similarities regarding two thresholds, $Th_s$ and $Th_m$. More precisely, a split for an object is confirmed when one or several components of it have the accumulated similarities to the rest of its components smaller than $Th_s$.

Then a new object node is created in the tree for the split components. Similarly, a merge is confirmed among objects when they are spatially connected and the similarities among those objects are larger than $Th_m$. Fig. 4.9 shows an example of object merge and split. In Fig. 4.9(a), the red component splits from its parent object and a new object marked in blue is created. In Fig. 4.9(b), the blue object is merged with the green object, since they are physically connected and the accumulated similarity between them is larger than $Th_m$. The components of these two objects are all connected to the one with larger size (the green object) while the one with smaller size (the blue object) is removed from the hierarchy.

### 4.5.3 Over Segmentation

The over-segmentation level in our hierarchical structure is employed to tackle the temporal consistency problem. In our approach, we over-segment components into segments to ensure that correct correspondences can be made between segments in the current hierarchy and the blobs detected in the next frame. Since the number of segments strongly affects the complexity of the task of building temporal correspondences, it is inappropriate to employ over-segmentation methods like super-voxels [PASW13], which generate a large number of segments. In this case, we propose a normalized cut based over-segmentation method working with the super-voxel graph of a component. In this graph, each node represents a super-voxel, and each edge is weighted by a measure of compactness between the two super-voxels it connects. We assume that any split will gradually reduce the compactness of the connections in the graph. So we perform a normalized cut iteratively in this graph to generate sub-graphs which are less compactly connected, that is, anticipating possible splits in the next frame.

For a pair of connected super-voxels $v_i$ and $v_j$ in graph $G_c$, we first define the touching points $TP_{i,j}$ between them as the points in one point cloud with the closest Euclidean distance to the points in the other point cloud, smaller than a threshold $Th_t$. Then, the con-

nection compactness $CC(v_i, v_j)$ between $v_i$ and $v_j$ is defined as the percentage of touching points between them.

$$CC(v_i, v_j) = \frac{NoP(TP_{i,j})}{NoP(C_{v_i}) + NoP(C_{v_j})} \tag{4.12}$$

A normalized min cut [SM00] is performed on the graph iteratively, thus creating one segment node in each iteration until the cut cost is larger than a threshold $Th_c$.

## 4.6 Experiments

We first evaluate the proposed method on two datasets for RGB-D video segmentation: the RGB-D video foreground segmentation dataset [FXL17] and the Human Manipulation dataset [PSPK14]. Apart from that, we also test our approach on 3 sequences provided in [HDT15] for comparison, and some other sequences without ground truth labelling for additional qualitative results. Ablation experiments are made in Section 4.6.3 on Human Manipulation dataset, in order to investigate the effect of Dynamic Management of Merge and Split (DMMS). After that, experiments in Section 4.6.3 are made to study the effect of fully connected CRF and super-voxel representation in the proposed approach. Experiments in Section 4.6.4 aims at evaluating our approach based on different graph building methods. Finally, we show the computational cost and some implementation details of the proposed approach in Section 4.6.5 and 4.6.6.

Note that all the RGB-D videos used in our experiments have the same resolution (640 by 480 for both color images and depth maps), and the voxel grid used for building the super-voxel graph is $1cm^3$.

| Name | nFrames | [FXL17] | ours |
|---|---|---|---|
| basketball2.2 | 40 | 42.60 | **55.13** |
| bdog.occ2 | 20 | 59.24 | **78.98** |
| br.occ.0 | 34 | 50.40 | **78.15** |
| child.no1 | 47 | 54.34 | **84.26** |
| dog.no.1 | 20 | 48.81 | **66.39** |
| studentcenter2.1 | 23 | 20.34 | **58.81** |
| toy.car.no | 35 | 38.51 | **62.19** |
| toy.green.occ | 31 | 64.66 | **78.83** |
| toy.wg.occ | 56 | **86.45** | 72.07 |
| tracking4 | 41 | 56.22 | **89.67** |
| walking.no.occ | 23 | 61.04 | **69.19** |
| zcup.move.1 | 36 | 64.07 | **79.96** |
| average | 34 | 53.90 | **72.80** |

Table 4.1: IOU scores for 12 sequences in RGB-D video foreground segmentation dataset reported in [FXL17] and for our method

### 4.6.1 Comparison Experiments on RGB-D Video Foreground Segmentation Dataset

The RGB-D video foreground segmentation dataset [FXL17] contains 12 RGB-D sequences captured in 7 different types of scenes with multiple objects. The first two columns of Table 4.1 specify the name of the 12 sequences and the corresponding number of frames. Challenges in these sequences are scenes with occlusions, interactions between objects, fast moving objects and camera movement. The ground truth labeled at pixel level for multiple objects is given for one out of every 5 frames. The authors also provide the results of their method in this database, which is based on the selection through graph optimization within a pool of object proposals using RGB-D data [FXL17]. We compare our method with the results provided in this dataset by employing the average Intersection Over Union (IOU)

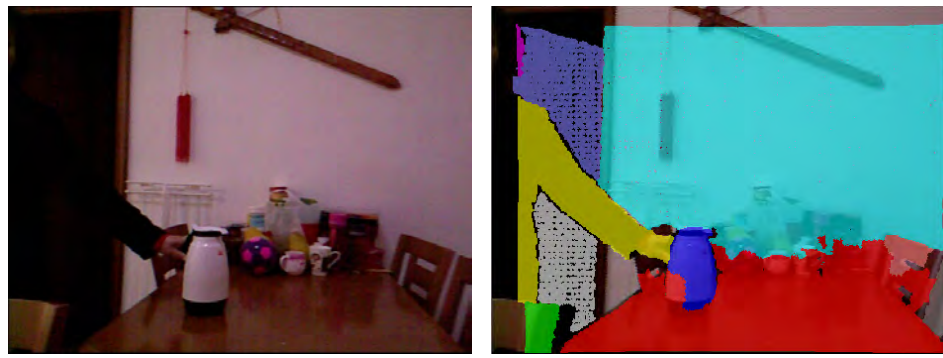

Figure 4.10: Qualitative results in RGB-D video foreground segmentation dataset

[FXZ+15] to measure the segmentation performance:

$$IOU = \frac{1}{M_o} \sum_{j=1}^{M_o} \max_i \frac{GT_j \cap R_i}{GT_j \cup R_i} \tag{4.13}$$

For each frame, $M_o$ stands for the number of objects labeled in the ground truth, $GT_j$ is the ground truth for object $j$ and $R_i$ represents the object proposals in the frame. Table 4.1 compares the IOU scores of the segmentation results of our approach with those obtained in [FXL17], while Fig. 4.10 shows some qualitative results from both methods. In [FXL17], the authors present the comparison between a number of other methods and their method, which achieves the best results in the RGB-D video foreground segmentation dataset. Our approach achieves better average IOU score: 72.80% compared to 53.90% in [FXL17] over the 12 sequences, as well as better average IOU in almost all sequences except "toy.wg.occ". Sequence "toy.wg.occ" involves very thin threads hanging two waving toys, where the 3D connectivity analysis are highly affected due to the low quality of the depth image on the very thin object parts. The proposed approach mainly based on sptaio-temporal connectivity analysis shows its drawbacks compared to the method proposed in [FXL17].

In Fig. 4.10, we only show the object proposals obtained with our approach which are related to the objects labeled in the ground truth to be able to compare with the method in [FXL17]. In fact, our approach segments the whole point cloud of the scene and obtains all foreground objects and their supporting planes, which are labeled accordingly as shown



(a) (b)

Figure 4.11: An example of the segmentation result in our method: (a) a color image (b) related segmentation mask

in Fig. 4.11.

*Application: Selection of Significant Objects.* In this experiment, we aim to select the significant objects from the object proposals obtained by our method along a sequence. This is usually achieved by evaluating the importance of object proposals along time based on some object attributes. In [FXL17], Fu et al. propose to select significant objects from a pool of object proposals through graph optimization, in which objectness, motion, RGB-D video saliency are involved to evaluate the importance of each object proposal. In our case, we employ the Frobenius norm of optical flow gradients as presented in Eq. 4.14 [ZJS13], which is the same motion term used in [FXL17], to represent the importance of an object proposal. In Eq. 4.14, $U(u, v)$ is the forward optical flow of a frame, $u_x$, $u_y$ and $v_x$, $v_y$ are the optical flow gradient in $x$ and $y$ direction respectively.

$$\|U_x\|_F = \left\| \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \right\|_F = \sqrt{u_x^2 + u_y^2 + v_x^2 + v_y^2} \tag{4.14}$$

Then, the evaluation is simply performed by averaging the importance of an object proposal in the history in each frame and selecting $k$ objects with higher importance along the sequence, since the temporal correspondences are made for all object proposals when they


Figure 4.12: Qualitative results of significant objects selection in RGB-D video foreground segmentation dataset

are generated in our approach. We compare our approach with [FXL17] on the RGB-D video foreground segmentation dataset. The qualitative comparison in Fig. 4.12 shows better results than [FXL17]. Our approach generates less but more accurate object proposals in each frame, which allows the system to establish object proposal correspondences online and simplifies the significant object selection problem.

### 4.6.2 Comparison Experiments for Sequences in [HDT15]

For comparison, we employ 3 more sequences proposed in [HDT15] and perform our approach against the Adaptive Surface Models based 3D Segmentation method (ASMS) in [HDT15]. Table 4.2 shows a quantitative comparison between our approach and ASMS in these 3 sequences. Sequence 1 contains a scenario of a human hand rolling a green ball forward and then backward with the fingers. Sequence 2 involves a robot arm grasping a paper roll and moving it to a new position. Sequence 3 describes a scenario in which a human hand enters and leaves the scene, displacing the objects rapidly. We evaluate the segmentation result by global accuracy. Global accuracy counts the percentage of pixels which are correctly labeled with respect to the ground truth labelling. The comparison results show that the proposed approach outperforms ASMS in all 3 sequences. It also illustrates one of the drawbacks in ASMS. In sequence 3, rapid object movement leaves little or no overlap of corresponding segments for ASMS to build temporal correspondences between objects

| Name | [HDT15] | ours |
|---|---|---|
| seq1 | 99.3 | 99.5 |
| seq2 | 82.1 | 86.0 |
| seq3 | 77.4 | 91.8 |
| average | 84.8 | 92.8 |

Table 4.2: Segmentation accuracy of our approach and the ASMS for the 3 sequences provided in [HDT15].

and update the object models. However, our method shows a higher robustness to cope with rapid movements, since the temporal correspondences are built by finding the global optimum of an assignment energy.

### 4.6.3 Ablation Experiments on Human Manipulation Dataset

#### Dynamic Management of Merges and Splits

To evaluate the improvements on performance when introducing Dynamic Management of Merges and Splits (DMMS), we employ 5 RGB-D sequences in the Human Manipulation dataset with the 3D point cloud ground truth labelling in consecutive frames provided in [LCP16]. Each of the sequences contains 201 frames of human manipulation actions which involve object interactions and self-occlusions/occlusions. The super-voxel based graph representation organizes the input point cloud with voxels in 3D producing a voxelized point cloud, while the ground truth is labeled in the original cloud. Therefore, we extend our segmentation result on the original cloud by simply finding k-nearest neighbors for each point on the original point cloud from our segmentation result. This allows using the majority voted label among k-nearest neighbors as the label for this point. Similarly, we employ the average intersection over union to measure and compare the performance of our approach with and without introducing DMMS in each frame.

Table 4.3 shows the segmentation performance of our approach in the 5 sequences with

| | without DMMS | with DMMS |
|---|---|---|
| Seq.1 | 93.95 | 96.87 |
| Seq.2 | 85.87 | 94.17 |
| Seq.3 | 91.16 | 96.63 |
| Seq.4 | 82.28 | 92.55 |
| Seq.5 | 88.39 | 90.46 |
| average | 88.33 | 94.14 |

Table 4.3: IOU scores for 5 sequences produced by our method without DMMS and with DMMS.

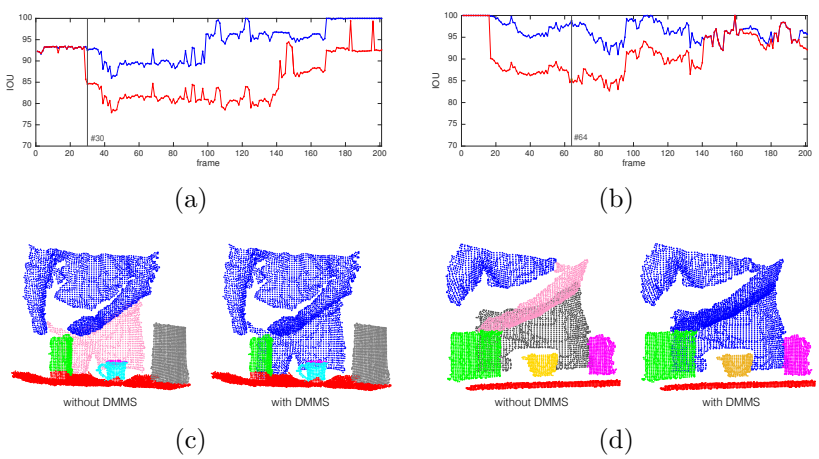



Figure 4.13: (a)-(b) present the IOU (vertical axis) per frame (horizontal) results for Seq 2-3. Red: our approach without DMMS, Blue: with DMMS.(c)-(d) present point cloud plots in frame 30 of Seq 2 and in frame 64 of Seq 3, object proposals are marked in different colors.

or without employing DMMS. In the case of "without DMMS", we do not maintain the similarities between corresponding nodes in the tree structures to update the object proposals along time. Table 4.3 exploiting DMMS provides improvements on segmentation performance in all 5 sequences (around 6% improvement in average IOU scores), which proves that DMMS contributes in the low level to the better segmentation of actual objects in the scene. In Fig. 4.13(a)-4.13(b), we present the IOU score per frame in 2 of the 5 sequences. The point cloud view in Fig. 4.13(c)-4.13(d) show that the torso of the human body splits

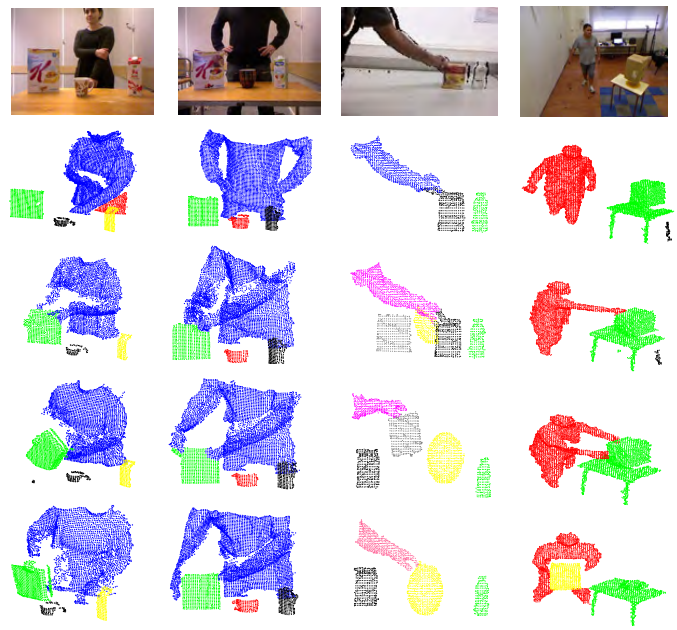


Figure 4.14: Qualitative results of the proposed method. Column 1-2: from human manipulation dataset in [PSPK14], Column 3: from data in [HDT15] and Column 4: from data recorded by ourselves.

into two parts (marked in blue and pink in the left point cloud in Fig. 4.13(c)) due to the self-occlusion, which leads to an improper segmentation in frame 30 of Seq 2. However, this is handled by DMMS which analyzes the correlations between those two parts in the history and maintains the similarity between them to produce a proper segmentation in the point cloud (shown as the right point cloud in Fig. 4.13(c)). Fig. 4.13(d) presents a similar situation in frame 64 of Seq 3, which also shows the importance of introducing DMMS. Fig. 4.14 presents more qualitative results produced by our approach. We manually remove some segments in the background for the clarity of the illustration. Each row in Fig. 4.14 shows the segmentation results in 4 frames of a sequence, which are uniformly sampled along the sequence. All these 4 examples illustrate that our approach handles the object instance segmentation task well in various scenes by only exploiting low level features and temporal

coherence. Low level features extracted from 3D point clouds are efficiently utilized in the proposed hierarchical structure to build temporal correspondences between object instances at different level, so that features for object instances can be temporally maintained for performing a good segmentation in the new frames.

### Fully Connected Conditional Random Field & Super-Voxel Representation

As introduced in Section 4.5.2, fully connected CRF is applied on super-voxels to achieve blob segmentation in our approach. To compare the impact of fully connected CRF and super-voxel representation, we employ 4 sequences in the Human Manipulation dataset [PSPK14] to evaluate the performance of three methods in blob segmentation, which are 1) traditional CRF on super-voxels (CRF), 2) fully connected CRF on super-voxels (S-FC-CRF), 3) fully connected CRF on pixels (P-FC-CRF). The 4 sequences employed in the experiment focus on scenes where a human interacts with multiple foreground objects, and contain over 2k frames with challenges like occlusion, fast moving objects and multi-objects interaction. The ground truth provides object labels for points on the point cloud in all the sequences. The evaluation metrics is used in the experiment is average Intersection over Union (IOU).

For this initial verification we perform each frame segmentation based on the ground truth object labeling of the previous frame. The temporal correspondences at the object level are made by comparing the ground truth on the previous frame and current frame, which allow to segment blobs corresponding to more than one object. Table 4.4 shows the segmentation performance of these three methods. S-FC-CRF achieves the best results in all sequences and obtains 2% improvement in total with respect to CRF based method. A 3.06% improvement is achieved by representing the raw pixels to super-voxels.

In the second experiment, we analyze the segmentation error for the whole sequence when we initialize the system with ground truth object labelling *only for the first frame* in

| Seq. No. | nFrames | CRF | P-FC-CRF | S-FC-CRF |
|---|---|---|---|---|
| 1 | 601 | 94.65 | 94.34 | **97.93** |
| 2 | 425 | 96.02 | 95.07 | **97.42** |
| 3 | 747 | 95.72 | 93.30 | **97.15** |
| 4 | 291 | 94.72 | 94.04 | **96.50** |
| average | 516 | 95.27 | 94.19 | **97.25** |

Table 4.4: Mean IOUs in 4 sequences of RGB-D videos produced by CRF, P-FC-CRF and S-CF-CRF

each of the 4 sequences. We analyze in Fig.4.15 the accumulated error in this situation. The P-FC-CRF method is not compared in this experiment due to its limited robustness to the accumulated segmentation error, which we justify in the following experiment. Fig.4.15 shows the segmentation result from CRF and S-FC-CRF (marked in red and blue respectively), in which we show the mean IOU in the vertical axis over the frames up to frame $t$ in the horizontal axis. The curve represents the trend of the segmentation performance. The S-FC-CRF based method keeps the segmentation performance at a higher level of mean IOU while also decaying slower than the CRF based method, which proves its stronger robustness with respect to the accumulated segmentation error.

In Fig.4.16(a), we show the performance of P-FC-CRF in the same manner than Fig.4.15. Compared to CRF and S-FC-CRF methods, it decays from 1 to 0.7 within around 30 frames, which also proves that the super-voxel representation employed in CRF and S-FC-CRF provides some robustness to the accumulated segmentation error.

We also evaluate the impact of employing different unary energies in S-FC-CRF. Specifically, we compare the unary energy defined only based on distance in our approach with a more complex unary energy which includes color, local surface normal and location in the S-FC-CRF method. Fig.4.16(b) illustrates that using the simple distance based unary energy achieves comparable segmentation performance (0.2% lower) than the more complex one.

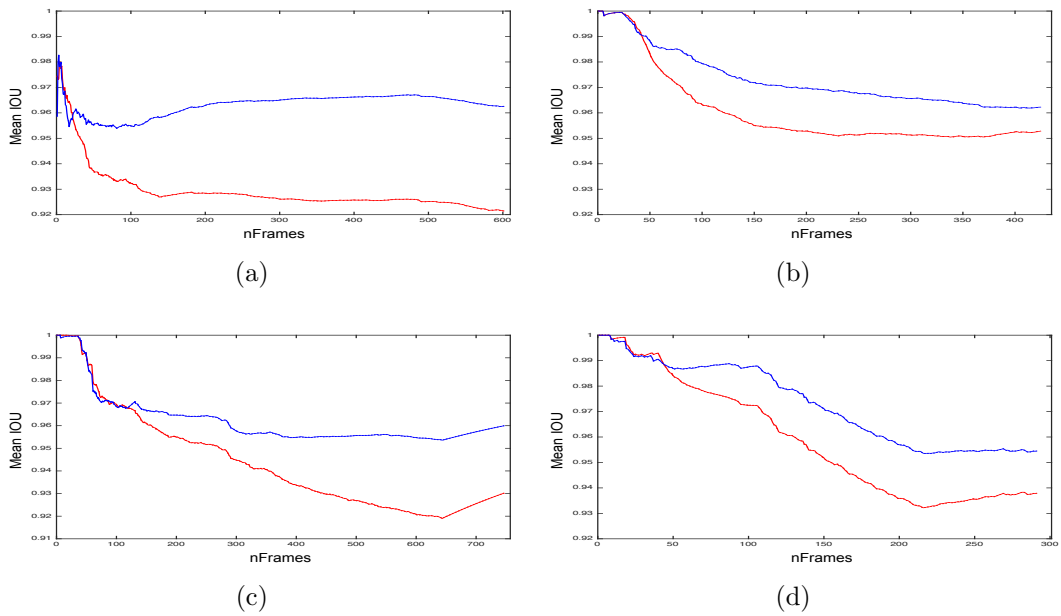



Figure 4.15: Segmentation performance shown as mean IOU (vertical axis) over $n$ frames (horizontal axis) in 4 different sequences. Red: method in [LCP16]. Blue: S-FC-CRF.

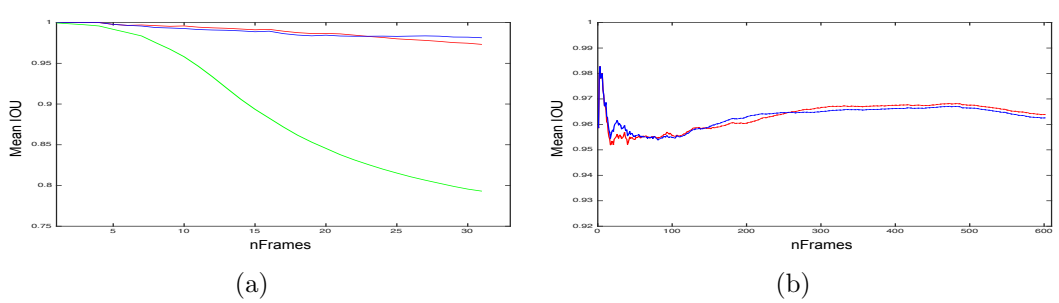



Figure 4.16: Segmentation performance verification: (a) on robustness to segmentation error for P-FC-CRF in green compared to CRF and S-FC-CRF in red and blue, (b) on employing different unary energy in S-FC-CRF, shown as mean IOU (vertical axis) over $n$ frames (horizontal axis). Blue: S-FC-CRF with unary energy defined based only on difference in location. RED: S-FC-CRF with unary defined based on difference in color, local surface normal and location.

### 4.6.4 Graph Building Methods Evaluation

As explained in Section 4.4.1, we employ the method proposed in [PKAW13, CSSPW14] to build the super-voxel graph for a point cloud. The edges on the graph represent the spatial connectivity of a point cloud. Our approach mainly focuses on analyzing point cloud connectivity spatio-temporally, which requires the graph building method to well describe point cloud connectivity in a scene. In this section, we compare two different graph building method, which are the super-voxel method (SV) [PKAW13, CSSPW14] explained in Section 4.4.1 and a restricted narrow band level set method (RNBLS) proposed in [SRHC13]. RNBLS employs level sets in RGB-D data to exploit connectivity over the depth surface. It expands and includes a set of new points on the point cloud with respect to the previous level set, under the constraints of proximity, density and color. In this manner, the constructed graph represents a point cloud while preserving its topology and boundary information.

We follow the experiment configuration in Section 4.6.3, where the ground truth of previous frame and the temporal correspondences are provided. We use 100 frames in each of the 4 sequences and apply the same segmentation method based on different graph building methods. Since the segmentation error only happens on the boundary of two objects at the attaching point, using IoU as the metrics usually do not reflect well the comparison between the two methods. Instead, we count the number of incorrectly segmented points compared to the ground truth. Fig.4.17 shows the comparison results in those 4 sequences. The graph representation obtained from SV outperforms the RNBLS method in all the 4 sequences.

### 4.6.5 Computational Cost

There are three main parts where the computational power is spent in our approach: the optimization for the multi-label assignment for the establishment of temporal correspondences, the fully connected CRF method used in the blob segmentation and the graph cut technique in the over-segmentation process. The main problem of approaching the temporal

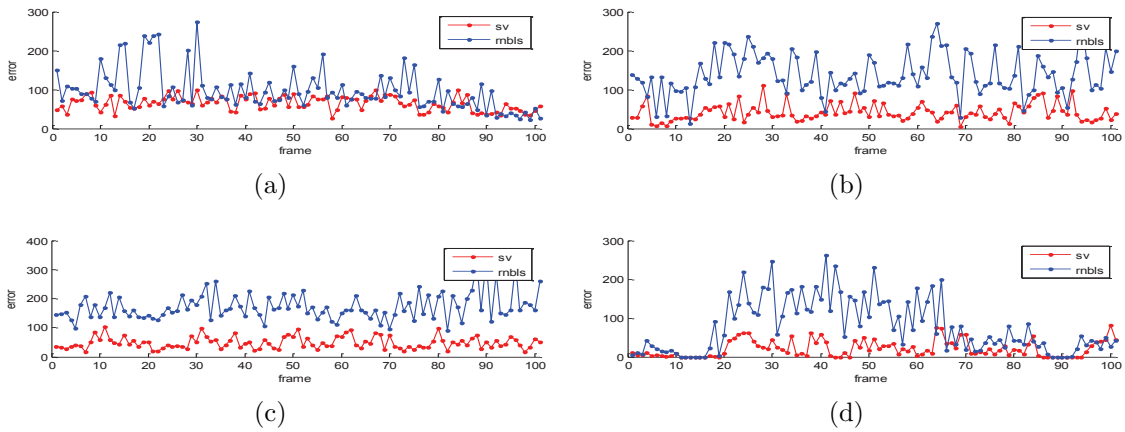



Figure 4.17: Quantitative results in error per frame for different sequences (a)-(d). Red point/line represents SV, blue point/line represents RNBLS

correspondences association by a multi-label assignment problem is the computation complexity. The problem scale increases exponentially with the number of labels. However, the number of labels is well controlled in our approach by finding a suitable over-segmentation level so that we can achieve the assignment task in a small scale while not leading to the temporal inconsistency problem. In the experiments, generally 20 segments are involved in the assignment task in each frame. In the fully connected CRF method [Kol11], the energy function is optimized using an efficient message passing method based on the mean fields approximation and high dimensional filtering, which makes the complexity of the approximated inference process sublinear in the number of the edges in the model. The graph cut technique used in our approach has the reported computation complexity $O\left(v^2 \cdot sqrt\left(e\right)\right)$ where $v$ stands for the number of vertices and $e$ the number of edges on the graph. Table 4.5 shows the run-time performance of the three main parts in our approach. All our experiments are performed on a machine with Mac OS and a 2.3GHz dual-core Intel Core i5 processor. The implementation of the proposed approach is developed by Matlab and C++ mixed programming. The 3 parts evaluated in Table 4.5 are developed in C++.

| | time | problem scale |
|---|---|---|
| Assignment Optimization | 2.3s | $\sim$ 15 segments, $\sim$ 8 blob labels |
| Over-Segmentation | 0.252s | $\sim$ 200 super-voxels $\sim$ 1500 edges |
| Blob Segmentation | 0.021s | $\sim$ 250 super-voxels $\sim$ 1800 edges |

Table 4.5: run-time performance of the proposed approach.

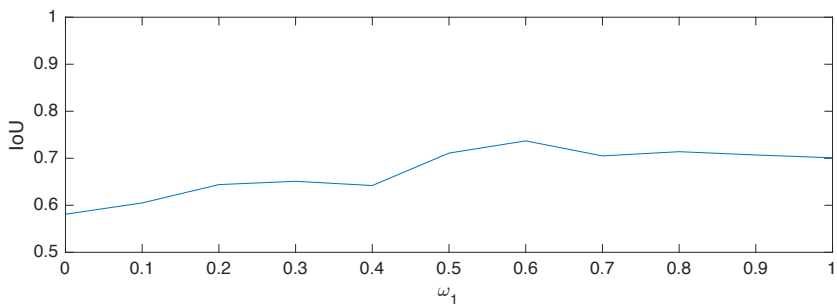



Figure 4.18: IoU scores for 20 validation images under different settings of $\omega_1$

### 4.6.6 Implementation Details

In this section we analyze the parameters used in the implementation of the proposed system. In the first place, the balance factors for the appearance and smoothness energies, $\omega_1$ and $\omega_2$ in Eq. 4.9, are learned from a small set of validation images. In practice, we select 20 examples, where blob segmentation is needed. For each example, we provide the ground truth object segmentation in the previous frame and build the previous hierarchy based on the ground truth object segmentation. Then we follow the proposed method to segment the current frame. The weights in Eq. 4.9 are then learned by searching for the best segmentation performance over these 20 training examples under different configuration of the weights. We set $\omega_1 + \omega_2 = 1$ and search for the best configuration of $\omega_1$ from 0 to 1 with step length 0.1. Fig. 4.18 shows the best segmentation result is obtained when $\omega_1$ is set to 0.6, and $\omega_2$ is set to 0.4. One advantage of dealing with actual 3D data is that the point cloud maintains the real size of objects in the scene, which provides a more clear physical

meaning for the related parameters. In our experiments, we fix $\sigma_\alpha$ and $\sigma_\gamma$ to 0.3 meter with respect to their physical significance. $\sigma_\beta$ is set to 13 following [Kol11]. $Th_s$ and $Th_m$ are the two parameters used for confirming the split and merge of an object by thresholding the component and object similarity. Thus, $Th_s$ and $Th_m$ are set to 1/3 and 2/3, which splits the similarity interval $[0, 1]$ into 3 zones (similar, neutral and not similar). $Th_t$ and $Th_c$ are the parameters used in over-segmentation. $Th_t$ is a 3D distance threshold specifying the touching points between two point clouds. $Th_c$ represents a graph cut cost threshold when performing normalized cut on the connection compactness graph. In our experiment, 0.07 meter is set for $Th_t$ and 0.03 is set for $Th_c$.

## 4.7 Conclusion

In this chapter, we have introduced a generic and temporally coherent 3D point cloud segmentation method for segmenting objects from generic scenes in RGB-D videos. We exploit temporal coherence by representing the generic point cloud segmentation in a single frame with a tree structure, and propagate it along time. Based on the hierarchical representation, we generate temporally coherent object segmentation at different scales of object-connectivity and establish reliable temporal correspondences between them. The behaviors of the temporally related nodes in the hierarchical structures built along time are further analyzed to produce better object segmentation.

We evaluate the performance of the proposed approach with the RGB-D video foreground segmentation dataset and the Human Manipulation data set, and compare it with state-of-the-art. Our approach generates a better segmentation result based on all low-level features available comparing to the state-of-the-art methods [FXL17, HDT15]. Due to the fact that only generic features are employed in the proposed approach, it is capable to segment generic objects in scenes and is free of initialization.

Chapter 5

# One-Shot Learning for Generic Instance Segmentation based on RGB-D stream data

## 5.1 Introduction

The performance of classical generic instance segmentation methods, such as [LCP18], is usually restricted to the discriminative power of the employed hand-crafted features. Those features are not representative enough to describe and distinguish different object instances when segmenting interacting object instances in generic scenes. On the other hand, Convolutional Neural Networks (CNNs) based semantic segmentation methods introduce a good representation for the predefined semantics, which are trained to extract robust features via networks with a huge number of parameters. Although the success of applying CNNs to semantic segmentation proves the strong representation capability of CNNs can be exploited on dense prediction tasks, it also shows some drawbacks. One of the major downsides of CNNs based approaches is their hunger for training data. In semantic segmentation, training data is prepared as manually labeled segmentation masks, in which labels in the mask represent different semantics. Preparing the training data for semantic segmentation requires large efforts on manual labeling due to the big necessity of training data. Besides, the idea

of semantic segmentation restricts to certain types of predefined semantics, which compromises its application to more generic scenes. From the perspective of generic segmentation, training data can hardly be prepared, since no semantics are predefined.

In video instance segmentation, methods proposed to detect/segment generic object instances, such as [EH10] and [LKG11], are usually employed as an object proposal generator. An offline temporal analysis is exploited, in order to search from a pool of object proposals within a frame along a video sequence, which, in consequence, restricts them to offline applications. On the other hand, model based generic instance segmentation methods, such as [HDT15, KLK14], usually employ online training techniques, where instance models are trained and updated along a video sequence. These approaches introduce a way to train instance models without predefined semantics. However, the models used in these approaches are usually simple, such as Gaussian models used in [KLK14] and quadratic functions in [HDT15], due to the small size of the training data.

In this chapter, we present a generic instance segmentation method which combines the advantages of the generic instance segmentation method introduced in Chap.4 [LCP18] and those of CNNs based semantic segmentation. That is the genericity in the generic instance segmentation method and the strong object representation power in CNNs, by exploiting the idea of one shot learning. We employ the classical generic instance segmentation method to discover object instances and build temporal correspondences based on all low level features. To represent the discovered object instances, we first train a CNN model offline for tracking generic object instances. Based on it, we fine-tune the tracking model online with the few examples of the discovered object instances, in order to obtain one CNN for each object instance to extract robust features. In that case, we can predict more accurately if a pixel belongs to the instance or not, based on the features extracted from CNNs rather than hand-crafted features used in [LCP18]. On the other hand, the genericity is also kept, since no prior information, such as initialization or predefined semantics, is introduced in the

proposed approach. Furthermore, in the experiments section we also evaluate the results obtained using the generic tracking CNN model trained offline, without object specific online fine-tuning. We observe that even these generic features outperform the hand-crafted ones, with a similar run-time performance.

## 5.2 Related Work

The most challenging part of the proposed approach is how to train the CNNs based system with very limited annotations. The deep architecture of CNNs provides a complex function with a large amount of parameters so that useful representations of high dimensional data can be learned. However, this advantage of CNNs becomes an obstacle in the training process when only few annotation is provided. In this case, the learned model is strongly over-fitted due to the large number of parameters and limited training data. To tackle the problem, we employ the idea of one shot learning. The key insight of one shot learning is that, rather than learning from scratch, one can take advantage of knowledge coming from a previously learned model and solve the new learning tasks using only one or few training samples.

One shot learning is an extreme case of transfer learning. Transfer learning is widely used for training CNNs in various tasks. For instance, [CPK$^+$16] trains a semantic segmentation network first on a image classification purpose using the large scale dataset ImageNet [DDS$^+$09] as the training data. Then, they take this pre-trained model as an initialization for a further training with a smaller set of training data for the semantic segmentation task. In [GDDM14], the authors also pre-train their object detection network with ImageNet on an image classification purpose.

One shot learning methods have also been developed for various tasks in the state-of-the-art, such as image recognition [VBL$^+$16, FFFP06] and gesture recognition [KH14]. More related to our approach, there are also one shot learning based approaches for video

object segmentation. In [CMPT+17], the authors present one shot object segmentation on video sequences, based on a fully-convolutional neural network architecture that is able to successively transfer generic semantic information, learned on ImageNet, to the task of foreground segmentation, and finally to learning the appearance of a single annotated object and segment the object in the following frames with the learned object model in the test sequence. Similarly, MaskTrack [PKB+17] learns to refine the detected mask of an object, by using the detections of the previous frame. The authors first synthesize the movement of an object mask between consecutive frames by performing affine transformation and non-rigid deformation to ground truth object masks in group of datasets. In this manner, the mask refinement network is generally trained off-line for generic objects in the group of datasets. Then, they fine-tune the network online for a specific object in a test sequence using only the ground truth mask provided in the first frame. One of the drawbacks of these approaches is that they require an accurate initialization for performing one shot learning on an object instance in the scene.

## 5.3 Classical Generic Instance Segmentation

In Chap.4, we have introduced a classical generic instance segmentation method $F$, which calculates the current segmentation $O_t$ in frame $t$ with point cloud $Ct$ obtained from the current RGB-D frame and the previous segmentation $O_{t-1}$, $F\left(C_t, O_{t-1}\right) \rightarrow O_t$. $O_t$ consists of different object instances $o_t^1, o_t^2 ... o_t^{M_o} \in O_t$, where $M_o$ denotes the number of objects in the scene. Since the temporal correspondences between object instances are made in $F$, we have the observed sequence of object instances in the history for each object instance $o_{1...t-1}^i$ before the segmentation in frame $t$ is obtained. To segment the current frame, the point cloud $C_t$ is first divided into blobs $b_t^1, b_t^2 ... b_t^{M_b}$ by analyzing the point cloud connectivity built on a super-voxel graph $G^t\left(v, e\right)$, in which $v$ represents super-voxels set and $e$ represents the edge set of the adjacency of super-voxels. The current blobs are then assigned to object

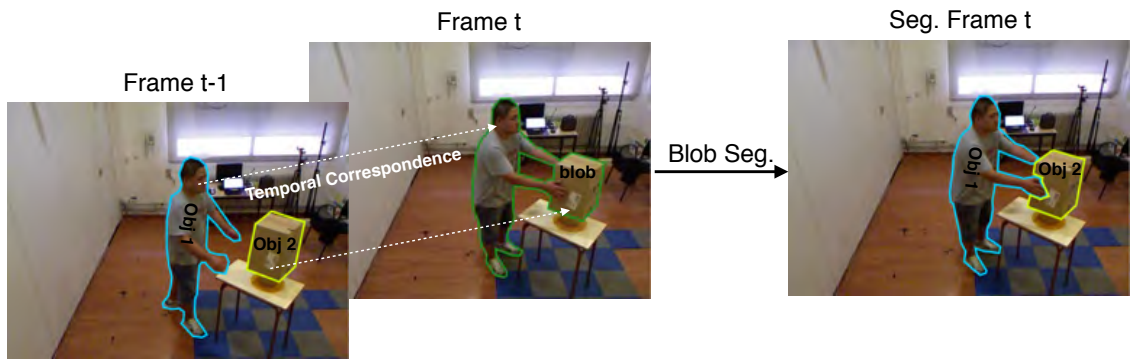



Figure 5.1: An example of blob segmentation in frame $t$ considering the temporally corresponded object instances in frame $t-1$.

labels from the previous frame via an optimization process. Blobs assigned to more than one object labels need Blob Segmentation. Fig.5.1 shows an example of a blob segmentation, in which a blob (the region with green boundary) in frame $t$ is segmented with respect to the object instances detected in frame $t-1$ (the region with blue and yellow boundary) and the temporal correspondence built between these two frames.

The segmentation for the first frame is simply done by first removing the plane-like point sets in the input point cloud, then searching for connected components on the super-voxel graph built on the residual point cloud. In this manner, isolated point sets are extracted from the input point cloud, which ideally corresponds to object instances in the scene.

In Chap.4, blob segmentation is achieved by labeling nodes on the graph of the blob with assigned object labels via a Fully Connected Conditional Random Field (FC-CRF) model. FC-CRF introduces an unary energy describing the degree of confidence that a super-voxel belongs to an object instance and a pairwise energy representing the degree of confidence that two super-voxels belong to the same object instance. Optimizing the energy function with the two energy terms provides the best labeling of the graph, which implicitly represents the segmentation of the blob. The unary energy for each node on the graph is defined based on low level features, such as 3D distance and color. As in [LCP16], we define

the unary energy for labelling node $v_i$ with object label $o_j$ as the mean distance between node $v_i$ in the current frame and the $k$-nearest nodes labeled by $o_j$ in the previous frame. This mean distance is computed comparing feature vectors which concatenate 3 components: color feature (color histogram in LAB color space), shape feature (local surface normal) and 3D position (3D coordinates of the node centroid). Details can be found in [LCP16]. These low level features are not always discriminative enough for well distinguishing/segmenting different object instances in a blob, which produces segmentation errors.

## 5.4 CNNs based Unary Energy Learning

To tackle the above mentioned problem, we propose to exploit CNNs to extract robust features for defining the unary energy in the blob segmentation task. In practice, we train one CNN model $N_i$ for each object instance based only on the few observations of that object instance in the history. The CNN $N_i$ extracts feature maps from the input data and outputs a 2 classes probability map via a softmax layer at the end of the CNN $N_i$. The probability map consists of probabilities that each pixel belongs to instance $i$ or not. For a super-voxel $v_j$, the probability is computed as the mean probability of pixels in $v_j$ (see Eq. 5.1). Then, we simply employ the probabilities of the super-voxels obtained from the CNN models of different object instances as the unary energy.

However, training CNNs with millions of parameters from scratch usually requires a large number of annotated data, in order to optimize the parameters for extracting robust representation of the input data. In our case, we only have few object instance observations in the history $o^i_{1...t-1}$ in frame $t$, which can be employed as training data. With limited number of annotated data, it is difficult to follow the training-from-scratch process. Thus, we follow the method proposed in [PKB+17] to perform one shot learning using the object instance observations in the history.

Given the segmentation of an object $o^i_{t-1}$ in frame $t-1$ and the input color image $I_t$ in

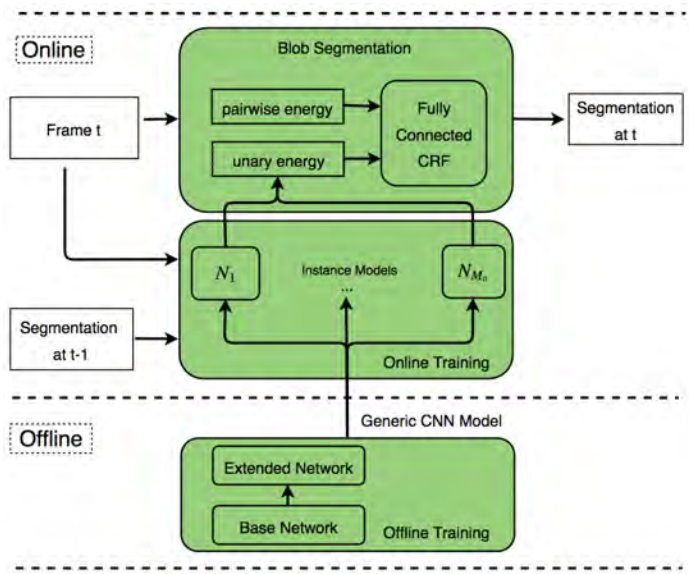



Figure 5.2: The schema of proposed approach.

frame $t$, our aim is to train a CNN $N_i\left(I_t, o_{t-1}^i\right) \rightarrow P_i^t$, where $P_i^t$ represents a probability map for object instance $o^i$ at time $t$. $P_i^t\left(x, y\right)$ stands for the output probability that the pixel $(x, y)$ on the input image $I_t$ belongs to object $o_t^i$ or not $(P\left(x, y\right) = \left[P_{o_t^i}\left(x, y\right), P_{\bar{o_t^i}}\left(x, y\right)\right])$. The CNN model generates the current object instance segmentation by refining the object instance segmentation $o_{t-1}^i$ in frame $t-1$ with respect to the current color image $I_t$. We formulate the CNNs based unary energy in Eq. 5.1 as:

$$\mu_{v_j}\left(i\right) = \frac{1}{M_{v_j}} \sum_{\forall(x,y)\in v_j} P_i(x, y) \tag{5.1}$$

where $M_{v_j}$ stands for the number of pixel contained in super-voxel $v_j$. Note that, in Eq. 5.1, we omit the notation $t$ for conciseness.

We employ two steps to achieve the training process: the offline training and online training step. In the offline training step, a base network is first employed to learn the generic attributes in an image classification task. Then, we extend the base network to learn a generic notion of how to segment an object instance taking a color image and a mask in

the previous frame as the input. In the online training step, we specify the extended network to a specific object instance by fine-tuning the the generic model obtained in the previous step, using only the few observations of the object instance in a sequence. Fig.5.2 shows the schema of the proposed approach.

### 5.4.1 Offline Training

A VGG network [SZ14] is used as our base network and is pre-trained on ImageNet [DDS+09] for an image classification task, which has proven to be a very good initialization in other tasks [CPK+16, GDDM14]. Although the network is not capable of performing image segmentation, it provides generic attributes in the network, which can be further specified to tackle other tasks.

The network is then extended to cope with the segmentation task. We follow DeepLab-ASPP [CPK+16], which replaces the fully connected layers in VGG network with atrous upsampling layers to achieve dense classification in a semantic segmentation task. DeepLab-ASPP is selected due to its outstanding performance in semantic segmentation. Then, we extend the network to allow an extra mask channel in the input. The extra mask channel is meant to provide an estimation of the visible area of the object in the current frame, its approximate location and shape. We can then train the extended network to output an accurate segmentation of the object instance, given as input the current image and a rough estimate of the object mask. To simulate the noise of the previous frame output, during offline training, we generate input masks by deforming the annotations using affine transformation as well as non-rigid deformations via thin-plate splines [Boo89], followed by a coarsening step (dilation morphological operation) to remove details of the object contour. We apply this data generation procedure over a dataset of $\sim 10^4$ images containing diverse object instances. The affine transformations and non-rigid deformations aim at modelling the expected motion of an object between two frames. The coarsening permits us to generate

training samples that resemble the test time data, simulating the blobby shape of the output mask given from the previous frame by the extended network, such that the network is trained to produce accurate output masks from a rough estimation from previous frame. These two ingredients make the estimation more robust to noisy segmentation estimates while helping to avoid accumulation of errors from the preceding frames.

### 5.4.2 Online Training

The offline training provides the extended network the ability to refine a roughly estimated mask of a generic object instance (e.g. the instance mask in the previous frame) to a segmentation of the object instance. In the case of a particular sequence, we fine-tune the extended network, in order to adapt it to the specific object instance based on the few observation of this object instance in the history.

Given the observations of an object instance $o^i_{1...t-1}, i \in \{1...M_o\}$ and the images $I_{1...t-1}$, we obtain $t-2$ training data, each of which contains $\left\langle o^i_{j-1}, I_j, o^i_j \right\rangle, j \in \{1...t-1\}$. Apart from this, we also perform data augmentation for the $t-1$ observations following the data generation method introduced in Section 5.4.1, in which we randomly generate $o^i_{j-1}$ for $\left\langle I_j, o^i_j \right\rangle$ by applying affine transform and non-rigid deformation. The extended model is fine-tuned based on these training data, in order to learn the appearance of a specific object instance and segment it in the current frame.

### 5.4.3 Training Details

Following the descriptions in previous subsections, we provide the training details of our network regarding the offline and online training strategies.

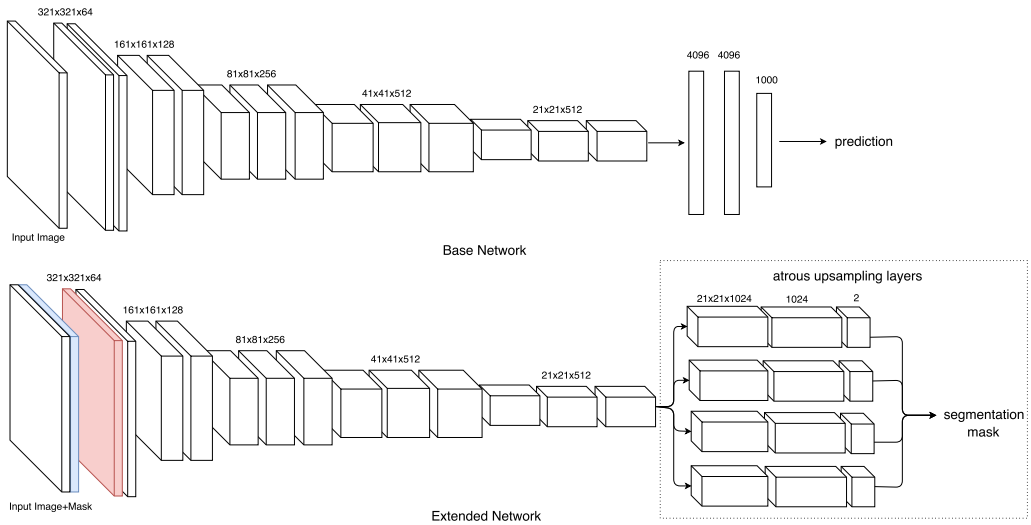



Figure 5.3: The architectures of the base network and extened network.

### Network Architecture

The base network follows the architecture of VGG network [SZ14]. VGG network employs 5 groups of convolutional layers with kernel size $3*3$ to extract robust features from an input image. Following each group of convolutional layers, a max pooling layer is provided to downsample the internal feature maps, so that the features can be extracted more globally in the following convolutional layers. Similarly, the extended network follows the architecture of DeepLab-ASPP which shares the same features network than VGG network [SZ14] and substitutes the fully connected layers in the VGG network with atrous upsampling layers. These atrous upsampling layers perform Atrous Spatial Pyramid Pooling (ASPP) on the feature maps to achieve the dense classification task in semantic segmentation. Following [PKB$^+$17], we extended DeepLab-ASPP to allow an extra mask channel in an input (denoted blue input channel in the extended network in Fig.5.3) by adding another channel in the filters of the first convolutional layer. Fig.5.3 shows an illustration of the architecture of the base network and extended. Note that the pooling layers are not shown in the figure for conciseness.

### Offline Training

The extended network is initialized from a base network pre-trained on ImageNet for an image classification task. For the added channel in filters of the first convolutional layer (see the red layer in the extended network in Fig.5.3) and atrous upsampling layers, we use Gaussian initialization. The training data used in the offline training process is generated from serveral datasets [CMH+15, LHK+14, ME10, SYXJ16] by performing affine transformation and thin-plate splines [Boo89]. That is to say, for each object mask $o$ on image $I$, we generate transformed and deformed masks of $o$, which forms several offline training samples. For affine transformation, we consider random scaling ($\pm 5\%$ of object size), translation ($\pm 10\%$ shift) and rotation ($\pm 10^{\circ}$). For deformation, we use 5 control points and randomly shift them within $\pm 10\%$ margin of the original object mask. Next, the mask is coarsened using dilation operation with 5 pixel radius. This mask deformation procedure is applied over all object instances in the training set. For each image two different masks are generated.

Since the extended network has identical parameters than the one proposed in [CPK+16], we follow its configuration for the hyper-parameters in our offline training process. In practice, we use Stochastic Gradient Descent (SGD) with mini-batches of 10 images and a polynomial learning policy with initial learning rate of 0.001. The momentum and weight decay are set to 0.9 and 0.0005, respectively. The network is trained for 20k iterations.

### Online Training

For online adaptation, we fine-tune the model previously trained offline for 200 iterations with training samples generated from the few observations in the history. We augment the few observations by image flipping and rotations as well as by deforming the annotated masks for an extra channel via affine and non-rigid deformations with the same parameters as for the offline training. This results in an augmented set of $\sim 10^3$ training images. The

network is trained with the same learning parameters as for offline training, fine-tuning all convolutional layers.

## 5.5 Experiment

In this section, we report the experiment results in the RGB-D video foreground segmentation dataset [FXL17] comparing with the classical generic instance segmentation (GIS) method introduced in Chap 4. The RGB-D video foreground segmentation dataset [FXL17] contains 12 RGB-D sequences captured in 7 different types of scenes with multiple objects. Since blob segmentation is needed only when objects interact with each other (physically attached), we perform both the CNN based generic instance segmentation (CNN+GIS) and the classical GIS on all the sequences, but evaluation is only made in frames which involve object interactions. We keep the evaluation metrics used by [LCP18] in the experiment as mean Intersection over Union (mIoU). Fig.5.4 shows some comparison results, in which results from CNN+GIS are shown in the first row and results from GIS in the second row. CNN+GIS obtains clearly improved segmentation results than GIS due to the better defined unary energy (see the better object boundaries obtained in CNN+GIS). A quantitative comparison is also made on this dataset, shown in Table 5.1. Apart from GIS and CNN+GIS, we introduce a comparison to CNN+GIS without performing online training (CNN+GIS-OT). CNN+GIS obtains around 6% higher mIoU than GIS, whereas CNN+GIS-OT also outperforms GIS with around 2% higher mIoU. To fully exploit the RGB-D data, we have also explored the possibility to incorporate the depth map as an extra input channel in CNN+GIS, however no improvement is observed, while the complexity is increased.

Table 5.2 shows average time spent for building the unary energy in GIS, CNN+GIS and CNN+GIS-OT in one blob segmentation respectively. Although CNN+GIS outperforms GIS in mIoU, the computational complexity is higher than GIS. With a trade-off in accuracy, the computation complexity of CNN+GIS can be decreased by eliminating online training


Figure 5.4: Examples of qualitative results from CNN+GIS in the first row and GIS in the second row.

| | mIoU |
|---|---|
| GIS | 67.2 |
| CNN+GIS | 73.5 |
| CNN+GIS-OT | 69.1 |

Table 5.1: Quanitative comparision between CNN+GIS and GIS

process or reducing the online training samples to obtain the expected run-time performance in the applications. The implementation for CNNs in all the experiments are developed under Caffe [JSD+14]. They are trained and tested with one Titan X graphics card with 12GB memory.

| | time |
|---|---|
| GIS | 0.07s |
| CNN+GIS | 12s |
| CNN+GIS-OT | 0.09s |

Table 5.2: Run-time performance of building the unary energy in CNN+GIS and GIS

## 5.6 Conclusion

In this chapter, we have presented a method which combines the strong object representation power in CNN based semantic segmentation methods and the genericity in the generic instance segmentation method introduced in Chap.4 [LCP18], and applied the combined approach to solve an instance segmentation problem. We verify the feasibility of employing one-shot learning method to model object instances with very few examples discovered by the generic object instance segmentation (GIS) method. The experiment results illustrate that an improved segmentation performance can be obtained by combining those two methods. On the other hand, instance independent learned features for tracking obtain a better result than hand-crafted features based on color, shape and 3D distance, with just a slight increase of the computational time. Features fine-tuned to the instance that is being tracked achieve the best results, but with a much higher run-time performance.

Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

This thesis is mainly divided into three parts, which focus on RGB-D based object segmentation problem from different perspectives.

In the first part (Chapter 3), the RGB-D based object segmentation problem is tackled when predefined semantics are available. In this case, the object segmentation problem is treated as a pixel level classification task, also known as semantic segmentation. One of the key points in semantic segmentation is to train a classifier to determine the class of each pixel. To adequately help training the classifier using RGB-D data, Convolutional Neural Networks (CNNs) have been employed to extract robust features. Different from the state-of-the-art works, such as [GGAM14] which directly take depth information as an extra input channel, we have proposed a more efficient and generic way to utilize depth information in the training process following the multi-task learning schema, in which we attempt to solve jointly the semantic segmentation and depth estimation task using a hybrid CNN. Due to the strong correlation between these two tasks, approaching them together with a hybrid network is mutually beneficial. On the other hand, the fact that depth information is not used as an input channel makes the system more general when facing situations when depth

data is not available in the testing phase. We also investigate the common attributes as well as the distinction for depth estimation and semantic segmentation and clarify how the two tasks help with each other in a hybrid system. Based on that, a novel hybrid CNN has been proposed in this thesis. The insight from our investigation is that the feature extraction process can not be simply merged due to difference in semantic segmentation and depth estimation. We find that splitting the global depth estimation from the common feature extraction process shared by the two tasks provides better performances in both tasks. This coincides with the fact that the global information needed in depth estimation has less impact in semantic segmentation. We have conducted experimental validation on different datasets including Cityscapes [COR+16] and SUN-RGBD [SLX15]. We have analyzed the performance of different hybrid architectures and shown the efficiency of the proposed architecture (HybridNet A2) in semantic segmentation and depth estimation. Comparison to other state-of-the-art hybrid architectures was also made, which also illustrates the comparable performance from the proposed hybrid architecture. The findings of this part of the thesis are presented in a published article and an article under preparation:

- D. Sanchez-Escobedo, X. Lin, Casas, J., and Pardàs, M., "HybridNet for Depth Estimation and Semantic Segmentation", in ICASSP 2018, In Press.

- X. Lin, D. Sanchez-Escobedo, Casas, J., and Pardàs M., "Depth Estimation and Semantic Segmentation from a Single RGB Image Using a Hybrid Convolutional Neural Network", Under Preparation.

In the second part (Chapter 4), the RGB-D based object segmentation problem is tackled in the situation that the temporal information is available while no predefined semantics are provided. In this case, the object segmentation problem is usually addressed in a more classical way. Tackling the object segmentation problem from this perspective brings naturally some advantages. For instance, these systems can handle generic object segmentation rather than restricted to some predefined objects, these systems do not require a large

amount of annotated data and the computational cost is lower. On the other hand, the lack of high level knowledge prevents them from bridging the gap between low level features and high level semantics. Thus, this part has been focused on the unsupervised generic instance segmentation problem in RGB-D stream data. We have presented a novel approach for 3D point cloud video (stream data) thoroughly exploiting the explicit geometry in RGB-D based on the analysis of 3D connectivity and compactness. To model the 3D point cloud data, we employ a graph representation based on super-voxels, which groups points on the point cloud into super-voxels locally and builds the 3D connectivity of a point cloud by constructing a super-voxel graph. Apart from that, a four layers hierarchical representation has been proposed for each frame to handle different level of object-connectivity, which is capable to cope with the object splits and merges. The temporal correspondences at different level of the hierarchies are established via an optimization process. With the temporal correspondences, we approach the segmentation problem by modelling it as Conditional Random Fields, where we minimize the energy function defined on current observation and the temporal correspondences using Graph-cuts. We have conducted experiments on Human Manipulation dataset [PSPK14] and RGB-D Video Foreground Segmentation Dataset [FXL17], as well as some other sequences from [HDT15] and sequences recorded by ourselves. We have analyzed our approach with ablation experiments, and compared our approach with the state-of-the-art approaches [FXL17, HDT15]. We have proved that the object level instance segmentation can be obtained by modelling the temporal correspondences of objects or object parts and analyzing their interactions along time in 3D point cloud videos. The findings of this part of the thesis were published in:

- X. Lin, Casas, J., and Pardàs, M., "Temporally Coherent 3D Point Cloud Video Segmentation in Generic Scene", IEEE Transactions on Image Processing, In Press.

- X. Lin, Casas, J., and Pardàs, M., "3D Point Cloud Segmentation Using a Fully Connected Conditional Random Field", in The 25th European Signal Processing Confer-

ence (EUSIPCO 2017), Kos island, Greece, 2017.

- X. Lin, Casas, J., and Pardàs, M., "3D Point Cloud Video Segmentation Based on Interaction Analysis", in ECCV 2016: Computer Vision – ECCV 2016, the Second International Workshop on Video Segmentation, Amsterdam, 2016, vol. III, 9915 vol., pp. 821 - 835.

- X. Lin, Casas, J., and Pardàs, M., "3D Point Cloud Segmentation Oriented to The Analysis of Interaction", in The 24th European Signal Processing Conference (EUSIPCO 2016), Budapest, Hungary, 2016.

In the third part (Chapter 5), we have proposed an approach which combines the advantages in both semantic segmentation and generic segmentation, where we discover generic object instances using the approach proposed in Chapter 4 and train for each discovered object instance a model using CNNs. In this case, the appearance of a discovered instance is modelled in a CNN rather than handcrafted features while the system retains its genericity, since no semantic information is introduced. The features extracted from CNNs are fed into the approach proposed in Chapter 4 to define the unary energy term in the energy function of CRFs, which generates better segmentation than using handcrafted features. Due to the limited number of examples of each discovered object instance, training an appearance model with CNNs is difficult. Thus, we employ the idea of one shot learning which performs knowledge transferring from the CNN models trained for general purposes where large annotated datasets are available. Based on that, a fine-tuning process is performed to adjust the model to a specific model of an instance. We have conducted experiments on RGB-D Video Foreground Segmentation dataset [FXL17] to validate our proposal. It proves that the learned instance appearance model can generate more robust features than handcrafted features used in Chapter 4, which improves the final segmentation performance. The findings of this part of the thesis are presented in an article to be submitted:

- X. Lin, Casas, J., and Pardàs, M., "One Shot Learning for Generic Instance Segmentation based on RGBD video", To be submitted.

Other publications which are not explicitly related to the topic of this thesis are listed as follows:

- P. A. Martínez, Lin, X., Castelán, M., Casas, J., and Arechavaleta, G., "A closed-loop approach for tracking a humanoid robot using particle filtering and depth data", Intelligent Service Robotics, vol. 10, no. 4, pp. 297–312, 2017.

- X. Lin, Casas, J., and Pardàs, M., "Time consistent estimation of End-effectors from RGB-D data", in Image and Video Technology: 7th Pacific-Rim Symposium, PSIVT 2015, Auckland, New Zealand, November 25-27, 2015, Revised Selected Papers, Cham, 2015, pp. 529-543.

Our work contributes to several projects:

- MALEGRA, TEC2016-75976-R, financed by the Spanish Ministerio de Economía, Industria y Competitividad and the European Regional Development Fund (ERDF)

- BIGGRAPH, TEC2013-43935-R, financed by the Spanish Ministerio de Economía y Competitividad and the European Regional Development Fund (ERDF)

## 6.2 Future Work

Although several segmentation tasks are discussed and addressed in this thesis, there are still some other interesting problems pending for the study in the future.

### 6.2.1 Learning Features from 3D Point Cloud using CNNs

In our approach proposed in Chapter 3, depth information, formed as depth maps, is considered as a reference to the target of training a multi-task hybrid CNN for semantic

segmentation and depth estimation. It will be interesting if we fully exploit the geometric information in depth maps by transforming them to 3D point clouds and learning feature extractors directly based on 3D point clouds. In this case, geometric features with higher discriminative power can be efficiently extracted in the real world scale for the pixel level classification. On the other hand, the loss function employed in the proposed hybrid network is a simple linear combination of the segmentation loss and depth estimation loss. By transforming depth maps to 3D point clouds, this loss could also be more naturally combined, such as defining the loss function on a 3D object bounding box estimation task which relates to both semantic segmentation and depth estimation.

### 6.2.2 Learning Features from Graph Representations using CNNs

In our approach proposed in Chapter 4, we construct graph representations for raw point clouds based on 3D connectivity, which provides a simplification of the original data. These graph representations are then exploited in an unsupervised way to build the temporal correspondences between frames and perform object level segmentation. In Chapter 5, we have tested the idea of combining the strong object representation power from CNNs with our generic instance segmentation approach using a one shot learning technique.

It will be interesting if we integrate CNNs based feature extractor learned from the graph representations, instead of color images. Since the graph representation is constructed based on 3D point clouds, it contains not only the color information in an image but also the geometric information of the scene. On the other hand, the simplicity of the graph representation also eases the task of feature learning compared to raw point clouds.

### 6.2.3 High Level Computer Vision Tasks

The segmentation approaches proposed in this thesis also build a solid foundation for high level computer vision tasks, such as robotic arm grasping, autonomous driving, human motion analysis, etc. These applications will also be investigated in our future work.

Appendix A

# Appendix to Chapter 3

## A.1 Fundamentals of Convolutional Neural Networks

While learning features has been a topic of interest for many years, considerable progress has been achieved in the last few years with the development of so-called deep learning methods [LB+95, KSH12, SZ14, HZRS16]. The key point to the success of deep learning methods is that they assemble the raw data hierarchically with deep neural networks, which provides multiple trainable networks stacked on top of each other showing different stage of the internal representation. On the other hand, the overall network is composed into a hierarchy of simple linear functions interleaved with some non-linearities, which makes the complex network trainable.

### A.1.1 Convolutional Neural Networks

The majority of the feature extractors follow a common framework consisting of a filter bank, a non-linear operation and finally a pooling operation. For instance, the Scale Invariant Feature Transform (SIFT) operator [Low04] applies oriented edge filters to a small

patch and determines the dominant orientation through a winner-take-all non-linear operation. Finally, the resulting sparse vectors are added (pooled) over a larger patch to form local orientation histograms [Far13]. Convolutional neural networks share the same schema, while employing multiple stages of feature extraction. Mathematically, a CNN with $K$ layers can be formulated as:

$$Y = f(X;\theta) = H_K \tag{A.1}$$

$$H_k = pool_k\left(act_k\left(W_k H_{k-1} + b_k\right)\right) \quad k \in \{1, ..., K-1\} \tag{A.2}$$

$$H_0 = X \tag{A.3}$$

where X is the input array of data, such as an image. $\theta = \{W, b\}$ represents all trainable parameters, in which $\{W_k, b_k\}$ shows the trainable weights and bias parameters at the $k$-th level. $act_k$ stands for a non-linear operation at layer $k$, while $pool_k$ is a pooling function at layer $k$. $H_k$ represents the obtained feature maps at layer $k$. The network is hierarchically structured, as the feature vector map $H_k$ at layer $k$ can be expressed as a function of the feature vector map obtained in the previous layer $H_{k-1}$ with respect to the parameters $\{W_k, b_k\}$ at layer $k$.

In CNNs, instead of performing full linear combinations of the previous feature vector map, the parameters at each layer are designed as a set of convolution kernels, where a feature map is obtained by sliding a convolution kernel on the previous feature vector map. In this manner, the number of the parameters is strongly reduced in a deep architecture, which makes the learning process (parameter estimation) easier. Concretely, if the input is a color image, each feature map would be a 2D array containing a color channel of the input image. Concatenating all feature maps at a layer forms the feature vector map. At the output, each feature map represents a particular feature extracted at all locations on the input. The output of a CNN is usually fed to a simple linear classifier.

From the mathematical description above, we can identify three key building blocks of

CNNs: 1) the convolutional layer serving as the filter bank in feature extractors, 2) the activation function providing non-linearities, 3) the pooling function.

### Convolutional Layer

The input of a convolutional layer is a feature vector map with $n_1$ 2D feature maps of size $n_2 \times n_3$. We denote the 2D feature map as $x_i$. The output of a convolutional layer is also a feature vector map consisting of $m_1$ feature maps of size $m_2 \times m_3$. The filter bank consists of trainable 2D filters (convolution kernel) $k_{i,j} \in k$ of size $l_1 \times l_2$, which computes the output feature map $y_j$ by taking feature maps $x$ as input via a 2D discrete convolution operator, that is $y_j = b_j + \sum_i k_{i,j} * x_i$, where $b_j$ is a trainable bias parameter. In this manner, each filter in the filter bank describes a feature extractor, which outputs a particular feature at every location of the input.

### Activation Function

The connected convolutional layers are linear combination in nature, that is to say, directly stacking convolutional layers is meaningless since the whole network is still equivalent to a single layer no matter how the layers are stacked. The activation function is a pointwise function which adds non-linearities in the obtained feature maps, which makes the stacking of convolutional layers meaningful.

The most widely used activation function is sigmoid function $\widehat{x} = \frac{1}{1+e^{-x}}$, where the output $\widehat{x}$ of the activation function is bounded between 0 to 1, which makes sure that the activation will not blow up in the hierarchically structured network. Similarly, tanh function $\widehat{x} = tanh\,(x) = \frac{2}{1+e^{-2x}} - 1$ is also employed in some approaches. However, these activation functions have the same problem that gradient of the activation function is vanishing towards either end of the activation function, which makes the parameter estimation slow or even stop in the training process.

The other type of activation function is called Rectified Learnear Unit (ReLU). ReLU function $\widehat{x} = max(0, x)$ outputs $x$ if $x$ is positive and 0 otherwise. ReLU can greatly accelerate the convergence of the learning process compared to the sigmoid/tanh functions due to its linear, non-saturating form. Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), ReLU can be implemented by simply thresholding a matrix of activations at zero. However, ReLU function can be fragile during training process. For example, a large gradient flowing through a ReLU function could cause the weights to update in such a way that the neuron will never activate on any data point again.

#### Pooling Function

The pooling function treats each feature map separately. In the simplest case, a pooling function averages values on a feature map over a neighborhood. The operation is performed with a step length larger than 1, called stride. This reduces the resolution of the feature map while introducing robustness to small variations in the location of the feature map. Other types of pooling function such as max pooling (use the max value within a neighborhood) or stochastic pooling (use a stochastic value within a neighborhood) are also used.

### A.1.2 Learning in Convolutional Neural Networks

Once the network $f$ has been chosen, the trainable parameters $\theta$ of the network can be learned with respect to a loss function $L$, in which we define how well the prediction from $f$ fits the target. We denote a training set of $N$ training samples $\{x^n, t^n\}$, where $x^n$ stands for an input data and $t^n$ a target value related to the training sample. Then the loss function can be represented as:

$$L\left(f; x, t, \theta\right) = \sum_{n} l\left(f\left(x^n; \theta\right), t^n\right) \tag{A.4}$$

where $l$ is a loss function capturing the per-sample loss to be optimized. $L$ represents the overall loss to be optimized.

Then the task of learning (parameter estimation) can be treated as a problem of minimizing the loss function $L$ over the training set $\{x^n, t^n\} \quad n \in \{1, ..., N\}$. This is usually tackled by performing a gradient descent procedure, since $f$ and $l$ are differentiable. In this case, the derivative of the loss function with respect to all the trainable parameters is computed using the back propagation algorithm, over the complete training set. Then the parameters are updated following the opposite direction of the gradient. However, it calculates the gradient with respect to the entire training set, which makes it extremely inefficient especially for training sets with a large of amount of training samples. This is typically tackled by introducing a stochastic approximation of the gradient, known as Stochastic Gradient Descent (SGD). SGD employs a subset of the entire training set as a batch and estimates the gradient based on the batch of data.

### A.1.3 Important General CNN architectures

With the fast development of CNNs in recent years, a large number of architectures are proposed to tackle different problems. Among them, there are some important architectures which are designed for advancement of general purposes, thus widely employed in approaches solving specific tasks. In this section, we summarize these important general architectures.

#### AlexNet

In 2012, Alex Krizhevsky et al [KSH12] created a large, deep convolutional neural network called AlexNet, which was used to win the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). It achieves an error rate of 15.4% compared to the second best approach achieved an error rate of 26.2%. This astounding improvement made deep convolutional architectures the hottest topic in the computer vision community. In his work, the network was made up of 5 convolutional layers, max-pooling layers and 3 fully connected layers. It was used for classification with 1000 categories in ImageNet. Figure A.1 shows

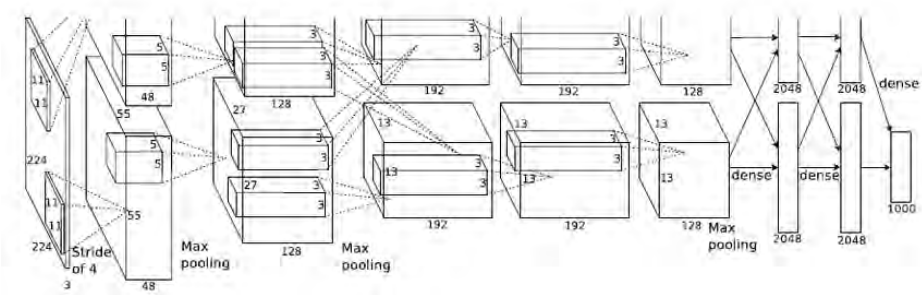



Figure A.1: The architecutre of AlexNet

the architecture of AlexNet.

### VGGNet

In 2014, Karen Simonyan et al [SZ14] proposed an architecture with a deeper CNN with 19 layers that strictly used 3x3 filters, along with 2x2 max pooling layers, called VGGNet. The use of only 3x3 sized filters decreases the number of parameters of the network, while keeping the effective receptive field by stacking more convolutional layers, since the combination of two 3x3 convolutional layers has an effective receptive field of 5x5. Figure A.2 shows the architecture of VGGNet.

### ResNet

Inspired by the idea that deeper architectures provides stronger representation, Kaiming He et al [HZRS16] proposed an extremely deep architecture, called ResNet. The work experimentally proves that a naive increase of layers in plain nets results in higher training and test error. To tackle this and create a very deep architecture, a residual block is proposed (see Figure A.3). The idea behind a residual block is that, instead of estimating the function of a group of convolutional layers $\mathscr{H}(x)$, one can estimate the residual function $\mathscr{F}(x) = \mathscr{H}(x) - x$ which corresponds to a slight change of the original input $x$ by using the residual block. In this manner, a 152 layers architecture was created (a stack of residual

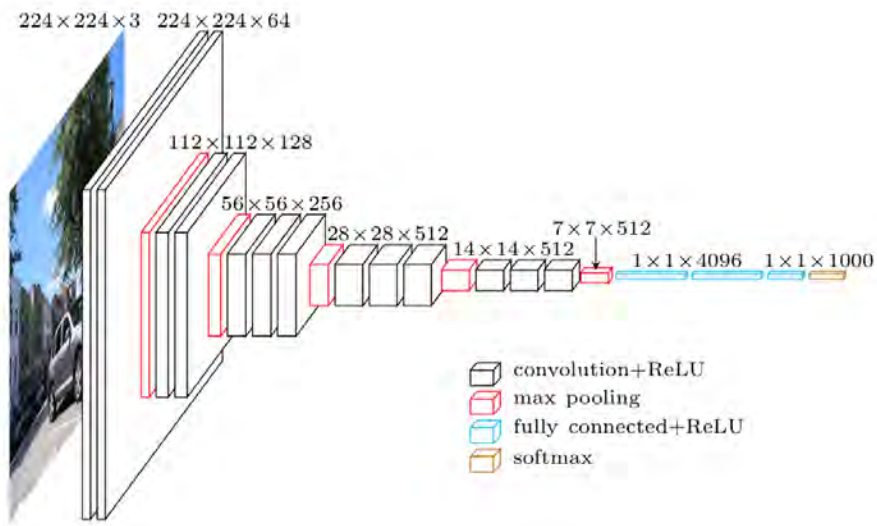



Figure A.2: The architecture of VGGNet

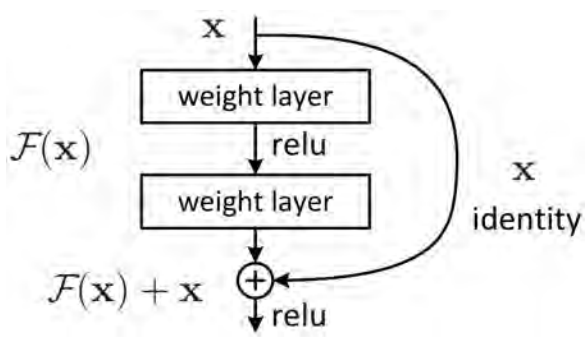



Figure A.3: The architecture of a residual block

blocks), which sets new records in classification, detection, and localization through this extremely deep architecture. ResNet won ILSVRC 2015 with an error rate of 3.6%.

Appendix B

# Appendix to Chapter 4

## B.1 Fundamentals of Conditional Random Fields

As explained in Section 2.2.4, graph based image segmentation approaches usually interpret the input data, such as pixels on an image or points on a point cloud, as vertices and edges in an undirected graph. Taking an image as an example, each vertex in the graph represents a pixel on the image, while the edge between two vertices shows their spatial relations. Normally, the spatial relation is defined based on 4-neighbors or 8-neighbors of a pixel. Full connectivity is also employed in some cases, in which each pixel is connected to every other pixel in the image. The goal of image segmentation is then converted to a task of obtaining structured labelling of graph nodes, where we label graph nodes with respect to the structure of the graph.

### B.1.1 Conditional Random Fields in Graph Based Image Segmentation

To model the image segmentation problem with a graph, we should consider output labels as inter-dependent, and explicitly model this inter-dependence. Conditional Random Field (CRF) [LMP01] is one of the methods employed to model the problem.

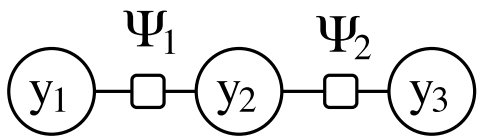



Figure B.1: An example of a factor graph

From the perspective of probabilistic graphical modeling, a probability distribution $p(Y)$ of interest can be represented by a product of factors of the form $\Psi_A(Y_A)$ where $A$ is an integer index of a factor in the factor set $\Lambda$. Each factor $\Psi_A$ depends only on a subset $Y_A$ of the variables $Y$. The reason for the term "graphical mode" is that the factorization can be compactly represented by means of a graph, called factor graph. Figure B.1 shows an example of a factor graph, where the variables $Y$ are depicted as round nodes while factors $\Psi$ are depicted as square nodes. The semantics of a factor graph is that the input of a factor function $\Psi_A$ is only related to the graph nodes which is connected to this factor.

A CRF is a discriminative undirected probabilistic graphical model [SM+12] that represents relationships between different variables. Consider a random field $Y$ defined over a set of variables. In an image segmentation problem, the domain of each variable is a set of labels. Consider also a set of variables $X$, representing features extracted from each pixel. $X$ ranges over the number of pixels on an image and $Y$ ranges over possible image labelings. CRFs model the mapping from the input variables $X$ to the output $Y$ via the conditional distribution $P(Y \mid X)$.

From the view of probabilistic graphical model, let $G(V, E)$ be an undirected graph, where $V$ is the set of nodes in the graph and each node corresponds to a variable $y_i \in Y$, and $E$ is the set of edges representing the spatial relation between $y$. A CRF assumes that the distribution of $P(Y \mid X)$ can be factorized according to a factor graph $G^f$ defined over $X$ and $Y$ for any value $x$ of $X$. Taking a 1D image as an example (shown in Figure B.2), graph $G$ on the left represents the spatial relations between the label of pixels $y$ on the 1D image. A factor graph $G^f$ defined based on $G$ shows how the distribution of $P(Y \mid X)$ can

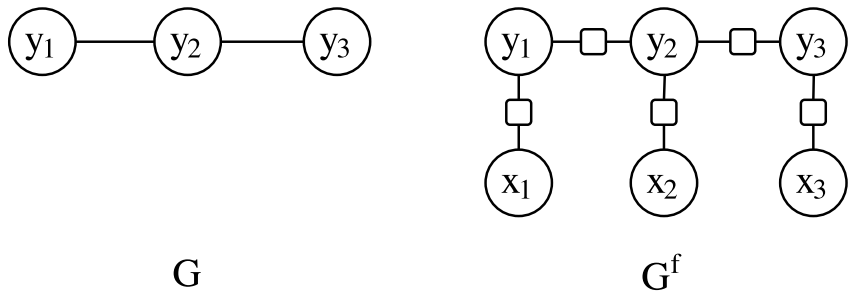



Figure B.2: A 1D image example of a factor graph of its CRF model

be factorized, where the squares represent the factors.

The reason that the factor graph of this 1D image is structured like Figure B.2 is that CRFs represent $y_i$ as a Markov random field, in which $y_i$ obeys the Markov Property when conditioned on $X$. That is, the conditional probability distribution of $y_i$ given its adjacent nodes is independent of the rest of the nodes in the graph. That is,

$$P\left(y_i \mid X, y_j, i \neq j\right) = P\left(y_i \mid X, y_j, j \in \mathbb{N}\left(i\right)\right) \tag{B.1}$$

where $x$ and $y$ denote the values assigned to variables $X$ and $Y$. $\mathbb{N}\left(i\right)$ is the adjacent nodes of $i$.

Thus, the conditional distribution $P\left(Y \mid X\right)$ of a CRF is presented as a normalized product of a set of non-negative potential functions, where each of the potential functions $\Psi$ stands for a factor on the factor graph, which is associated with a set of nodes $A$:

$$P\left(y \mid x\right) = \frac{1}{Z\left(x\right)} \prod_{A \in \Lambda} \Psi_A\left(y_A, x_A\right) \tag{B.2}$$

where $\Lambda$ stands for the factor set. $Z$ is a normalization factor which is also called partition function depending on the observed values of input variable $X$ and is defined as:

$$Z\left(x\right) = \sum_{y} \prod_{A \in \Lambda} \Psi_A\left(y_A, x_A\right) \tag{B.3}$$

It is also assumed that the conditional distribution over graph $G^f$ is an exponential family. Thus we require each potential function $\Psi_A$ to have the form:

$$\Psi_A = exp\left\{\sum_k \omega_{A_k} f_{A_k}\left(x_A, y_A\right)\right\} \tag{B.4}$$

where $\omega_{A_k}$ is the real value parameter for the set of feature function $\{f_{A_k}\}$ defined on potential $\Psi_A$.

The task of image segmentation (node labeling) using CRFs is to find the $y^*$ which maximizes the conditional probability $y^* = argmax_y P\left(y|x\right)$. To simplify the solution to an energy function, one can take the negative logarithm of the left hand side and right side of Equation B.2 and ignore the normalization constant $Z$. The problem of maximizing the conditional probability becomes an energy minimization problem. The form of the energy function depends on the definition of the set of feature function $\{f_{A_k}\}$. Normally, it consists of a unary term and a pairwise term. In this case, the energy function can be written as:

$$E\left(y \mid x\right) = \sum_i \mu\left(y_i \mid x\right) + \sum_{i,j} \rho\left(y_i, y_j \mid x\right) \tag{B.5}$$

where the unary energy describes how well a label fits a node on the graph given the observed data and the pairwise energy represents the extend that the labels of two nodes are pairwise smooth.

### B.1.2 Approximate Energy Minimization using Graph Cut

Due to different definitions of the unary and pairwise term, the energy function in Equation B.5 might be convex or not. The major difficulty with energy minimization lies in the enormous computational costs. Typically, these energy functions have many local minimum (i.e., they are non-convex). Worse still, the space of possible labelings has high dimension (equal to the number of pixels in an image segmentation task), which is many

thousands.

In an image segmentation task, the pairwise term is usually defined to preserve discontinuities on object boundaries. This makes it NP hard to find the global optimum of the energy function [BVZ01]. Due to the inefficiency of computing the global minimum, authors in [BVZ01] propose a fast approximate energy minimization method based on graph cut and $\alpha$ expansion. Given an initial labeling, the idea of $\alpha$ expansion is to expand a label (for instance *alpha*) to non-*alpha* labeled nodes, which reduce the energy respect to the energy function. It is also proved in [BVZ01] that an optimal $\alpha$ expansion move can be obtained by searching for the minimum cut on a graph constructed with respect to the unary and pairwise term defined in the energy function. In practice, $\alpha$ expansion moves are performed for each label in the label set in one round of minimization. The optimization process stops when all of the $\alpha$ expansion moves in one round could not reduce the energy in the energy function or the number of rounds reaches to a predefined number.

# Bibliography


[AM98] M Stella Atkins and Blair T Mackiewich. Fully automatic segmentation of the brain in mri. *IEEE transactions on medical imaging*, 17(1):98–107, 1998.

[AM17] Lotfi Abdi and Aref Meddeb. Driver information system: a combination of augmented reality and deep learning. In *Proceedings of the Symposium on Applied Computing*, pages 228–230. ACM, 2017.

[AMFM11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[APP+12] Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Wörgötter, and Babette Dellen. Depth-supported real-time video segmentation with the kinect. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 457–464. IEEE, 2012.

[APTB+14] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.

[ASS+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[BFC09] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[BMBM10] Lubomir Bourdev, Subhransu Maji, Thomas Brox, and Jitendra Malik. Detecting people using mutually consistent poselet activations. *Computer Vision–ECCV 2010*, pages 168–181, 2010.

[Boo89] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[BRE+17] David Ball, Patrick Ross, Andrew English, Peter Milani, Daniel Richards, Andrew Bate, Ben Upcroft, Gordon Wyeth, and Peter Corke. Farm workers of the future: Vision-based robotics for broad-acre agriculture. *IEEE Robotics & Automation Magazine*, 2017.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[CDF+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[CJSW01] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[CLP98] Chang Wen Chen, Jiebo Luo, and Kevin J Parker. Image segmentation via adaptive k-mean clustering and knowledge-based morphological operations with biomedical applications. *IEEE transactions on image processing*, 7(12):1673–1683, 1998.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[CMH+15] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

[CMPT+17] S. Caelles, K.K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[CP08] Gabriela Csurka and Florent Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, pages 1–10, 2008.

[CPK+16] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[CSSPW14] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[EF15] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[EH10] Ian Endres and Derek Hoiem. Category independent object proposals. In *European Conference on Computer Vision*, pages 575–588. Springer, 2010.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[Far13] Clément Farabet. *Towards real-time image understanding with convolutional networks*. PhD thesis, Université Paris-Est, 2013.

[FF56] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[FFFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[FGMR10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[FH04] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[FXL17] H Fu, D Xu, and S Lin. Object-based multiple foreground segmentation in rgbd video. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 2017.

[FXLL15] Huazhu Fu, Dong Xu, Stephen Lin, and Jiang Liu. Object-based rgbd image co-segmentation with mutex constraint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4428–4436, 2015.

[FXZ+15] Huazhu Fu, Dong Xu, Bao Zhang, Stephen Lin, and Rabab Kreidieh Ward. Object-based multiple foreground video co-segmentation via multi-state selection graph. *IEEE Transactions on Image Processing*, 24(11):3415–3424, 2015.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Gei08] Wilson S Geisler. Visual perception and the statistical properties of natural scenes. *Annu. Rev. Psychol.*, 59:167–192, 2008.

[GGAM14] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[GKHE10] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141–2148. IEEE, 2010.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[Gol10] E Bruce Goldstein. *Encyclopedia of perception*, volume 1. Sage, 2010.

[HBEC14] Steven Hickson, Stan Birchfield, Irfan Essa, and Henrik Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *CVPR2014*. IEEE Computer Society, 2014.

[HDT15] Farzad Husain, Babette Dellen, and Carme Torras. Consistent depth video segmentation using adaptive surface models. *Cybernetics, IEEE Transactions on*, 45(2):266–278, 2015.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[HH75] John A Hartigan and JA Hartigan. *Clustering algorithms*, volume 209. Wiley New York, 1975.

[HHRB11] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *Robot Soccer World Cup*, pages 306–317. Springer, 2011.

[Ho95] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Iva16] Bc Ján Ivanecký. Depth estimation by convolutional neural networks. Master thesis, Brno University of Technology, 2016.

[JKJ+13] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision*, pages 141–165. Springer, 2013.

[JLD03] Kan Jiang, Qing-Min Liao, and Sheng-Yang Dai. A novel white blood cell segmentation scheme using scale-space filtering and watershed clustering. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 5, pages 2820–2825. IEEE, 2003.

[Joh97] Andrew E Johnson. *Spin-images: a representation for 3-D surface matching*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 1997.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[KH14] Jakub Konecnỳ and Michal Hagara. One-shot-learning gesture recognition using hog-hof. *Journal of Machine Learning Research*, 15:2513–2532, 2014.

[KLK14] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Incremental object learning and robust tracking of multiple objects from rgb-d point set data. *Journal of Visual Communication and Image Representation*, 25(1):108–121, 2014.

[Kol11] Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2(3):4, 2011.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KT+09] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[LB+95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[LCP16] X. Lin, J. Casas, and M. Pardàs. 3d point cloud segmentation oriented to the analysis of interactions. In *The 24th European Signal Processing Conference (EUSIPCO 2016)*. Eurasip, 2016.

[LCP18] Xiao Lin, Josep R Casas, and Montse Pardàs. Temporally coherent 3d point cloud video segmentation in generic scenes. *IEEE Transactions on Image Processing*, 2018.

[LHK+14] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287, 2014.

[LKG11] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *2011 International Conference on Computer Vision*, pages 1995–2002. IEEE, 2011.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[ME10] Vida Movahedi and James H Elder. Design and perceptual validation of performance measures for salient object segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 49–56. IEEE, 2010.

[MKRVG15] Andelo Martinovic, Jan Knopp, Hayko Riemenschneider, and Luc Van Gool. 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2015.

[ML12] Tianyang Ma and Longin Jan Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 670–677. IEEE, 2012.

[MPK16] Arsalan Mousavian, Hamed Pirsiavash, and Jana Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 611–619. IEEE, 2016.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[NSF12] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[PASW13] Jeremie Papon, Andrey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2027–2034. IEEE, 2013.

[PCMY15] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

[PKAW13] Jeremie Papon, Tomas Kulvicius, Eren Erdal Aksoy, and Florentin Wörgötter. Point cloud video object segmentation using a persistent supervoxel world-model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3712–3718. IEEE, 2013.

[PKB+17] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017.

[PS13] Guillem Palou and Philippe Salembier. Hierarchical video representation with trajectory binary partition tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2099–2106, 2013.

[PSPK14] Alessandro Pieropan, Govind Salvi, Karl Pauwels, and Hedvig Kjellstrom. Audio-visual classification and detection of human manipulation actions. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3045–3052. IEEE, 2014.

[PTN09] Nils Plath, Marc Toussaint, and Shinichi Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 817–824. ACM, 2009.

[RB94] Daniel L Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. In *Advances in neural information processing systems*, pages 551–558, 1994.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[RHBB09] Radu Bogdan Rusu, Andreas Holzbach, Nico Blodow, and Michael Beetz. Fast geometric point labeling using conditional random fields. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 7–12. IEEE, 2009.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[RM00] Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1, 2):187–228, 2000.

[RM07] Xiaofeng Ren and Jitendra Malik. Tracking as repeated figure/ground segmentation. In *Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

[Rus09] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[SA11] Luciano Spinello and Kai O Arras. People detection in rgb-d data. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3838–3843. IEEE, 2011.

[SHKF12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012.

[SJC08] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[SKK16] Utsav Shah, Rishabh Khawad, and K Madhava Krishna. Deepfly: towards complete autonomous navigation of mavs with monocular camera. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, page 59. ACM, 2016.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SLX15] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[SM+12] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

[SRHC13] Xavier Suau, Javier Ruiz-Hidalgo, and Josep R Casas. Detecting end-effectors on 2.5 d data using geometric deformable models: Application to human pose estimation. *Computer Vision and Image Understanding*, 117(3):281–288, 2013.

[SSF14] Nathan Silberman, David Sontag, and Rob Fergus. Instance segmentation of indoor scenes using a coverage loss. In *European Conference on Computer Vision*, pages 616–631. Springer, Cham, 2014.

[SW97] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[SYRK+17] Jae Shin Yoon, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2176, 2017.

[SYXJ16] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):717–729, 2016.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TFNR12] David Tsai, Matthew Flagg, Atsushi Nakazawa, and James M Rehg. Motion coherent tracking using multi-label mrf optimization. *IJCV*, 100(2):190–202, 2012.

[TWZ+16] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*, 2016.

[UCFB16] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.

[VBL+16] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[VMS08] Veronica Vilaplana, Ferran Marques, and Philippe Salembier. Binary partition trees for object detection. *IEEE Transactions on Image Processing*, 17(11):2201–2216, 2008.

[WGB12] David Weikersdorfer, David Gossow, and Michael Beetz. Depth-adaptive superpixels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2087–2090. IEEE, 2012.

[Whi94] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[WL93] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

[WSL+15] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015.

[XCGN17] Yongchao Xu, Edwin Carlinet, Thierry Géraud, and Laurent Najman. Hierarchical segmentation using tree-based shape spaces. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):457–469, 2017.

[XOT13] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[YHRF12] Yi Yang, Sam Hallman, Deva Ramanan, and Charless C Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[Zah71] Charles T Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers*, 100(1):68–86, 1971.

[ZJS13] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, 2013.